

On Integer Programming and the Path-width of the Constraint Matrix

Fedor V. Fomin¹, Fahad Panolan¹, M. S. Ramanujan², and Saket Saurabh^{1,3}

¹Department of Informatics, University of Bergen, Norway.

{fomin|fahad.panolan}@ii.uib.no

²Technische Universität Wien, Vienna, Austria. ramanujan@ac.tuwien.ac.at

³The Institute of Mathematical Sciences, HBNI, Chennai, India. saket@imsc.res.in

Abstract

In the classic *Integer Programming* (IP) problem, the objective is to decide whether, for a given $m \times n$ matrix A and an m -vector $b = (b_1, \dots, b_m)$, there is a non-negative integer n -vector x such that $Ax = b$. Solving (IP) is an important step in numerous algorithms and it is important to obtain an understanding of the precise complexity of this problem as a function of natural parameters of the input.

Two significant results in this line of research are the pseudo-polynomial time algorithms for (IP) when the number of constraints is a constant [Papadimitriou, J. ACM 1981] and when the branch-width of the column-matroid corresponding to the constraint matrix is a constant [Cunningham and Geelen, IPCO 2007]. In this paper, we prove *matching* upper and lower bounds for (IP) when the *path-width* of the corresponding column-matroid is a constant. These lower bounds provide evidence that the algorithm of Cunningham and Geelen, are probably optimal. We also obtain a separate lower bound providing evidence that the algorithm of Papadimitriou is close to optimal.

1 Introduction

In the classic *Integer Programming* problem, the input is an $m \times n$ integer matrix A , and an m -vector $b = (b_1, \dots, b_m)$. The objective is to find a non-negative integer n -vector x (if one exists) such that $Ax = b$. Solving this problem, denoted by (IP) is an important step in numerous algorithms and it is important to obtain an understanding of the precise complexity of this problem as a function of natural parameters of the input.

In 1981, Papadimitriou [26] showed that (IP) is solvable in pseudo-polynomial time on instances for which the number of constraints m is a constant. His proof consists of two steps. The first step is combinatorial, showing that if the entries of A and b are from $\{0, \pm 1, \dots, \pm d\}$, and (IP) has a solution, then there is also a solution which is in $\{0, 1, \dots, n(md)^{2m+1}\}^n$. The second, algorithmic step shows that if (IP) has a solution with the maximum entry at most B , then the problem is solvable in time $\mathcal{O}((nB)^{m+1})$. In particular, when the matrix A happens to be non-negative, his algorithm for IP runs in time $\mathcal{O}((nd)^{m+1})$ where $d = \max\{b_1, \dots, b_m\}$. A natural question therefore is whether the algorithm of Papadimitriou can be improved significantly in general and in particular for the case when A is non-negative. Our first theorem provides a conditional lower bound indicating that any significant improvements are unlikely. To be precise, we prove the following theorem.

Theorem 1.1. *Unless the Exponential Time Hypothesis (ETH) fails, (IP) with $m \times n$ matrix A cannot be solved in time $n^{o(\frac{m}{\log m})} d^{o(m)}$, where $d = \max\{b_1, \dots, b_m\}$, even when the constraint matrix A is non-negative and each entry in any feasible solution is at most 2.*

ETH is the conjecture that 3-SAT cannot be solved in time $2^{o(n)}$ on n -variable formulas [21]. Due to Theorem 1.1, the simple dynamic programming algorithm [26] for (IP) when the maximum entry in a solution, as well as in the constraint matrix, is bounded, is already close to optimal. In fact, when the constraint matrix is *non-negative*, our lower bound asymptotically almost matches the $\mathcal{O}((nd)^{m+1})$ running time of Papadimitriou’s algorithm [26]. Hence, we conclude that obtaining a significant improvement over the algorithm of Papadimitriou for non-negative matrices is at least as hard as obtaining a sub-exponential time ($2^{o(n)}$) algorithm for 3-SAT. In fact, observe that based on the setting of the parameters m, d, n our lower bound rules out several interesting running times. For instance, if $m = \Theta(n)$ and $d = \mathcal{O}(1)$, we immediately get a $2^{o(n)}$ lower bound.

Continuing the quest for faster algorithms for (IP), Cunningham and Geelen [8] suggested a new approach for solving (IP) which utilizes a *branch decomposition* of the matrix A . They were motivated by the fact that the result of Papadimitriou can be interpreted as a result for matrices of constant *rank* and branch-width is a parameter which is upper bounded by rank plus one. Robertson and Seymour [28] introduced the notion of *branch decompositions* and the corresponding notion of branch-width for graphs and more generally for matroids. Branch decompositions have immense algorithmic significance because numerous NP-hard problems can be solved in polynomial time on graphs or matroids of constant branchwidth [7, 16, 19, 18]. For a matrix A , the *column-matroid* of A denotes the matroid whose elements are the columns of A and whose independent sets are precisely the linearly independent sets of columns of A . We postpone the formal definitions of branch decomposition and branch-width till the next section. For (IP) with a *non-negative* matrix A , Cunningham and Geelen [8] showed that when the branch-width of the column-matroid of A is constant, (IP) is solvable in pseudo-polynomial time.

Theorem 1.2 (Cunningham and Geelen [8]). *(IP) with non-negative $m \times n$ matrix A given together with a branch decomposition of its column matroid of width k , is solvable in time $\mathcal{O}((d+1)^{2k}mn + m^2n)$, where $d = \max\{b_1, \dots, b_m\}$.*

Upper Bounds	Lower bounds
$\mathcal{O}(nd)^{m+1}$ [26] (non-negative matrix A)	no $n^{o(\frac{m}{\log m})}d^{o(m)}$ algorithm under ETH (Theorem 1.1) (even for non-negative matrix A)
$\mathcal{O}((d+1)^{\text{pw}+1}mn + m^2n)$ (Theorem 1.3) (non-negative matrix A)	no $f(\text{pw})(d+1)^{(1-\epsilon)\text{pw}}(mn)^{\mathcal{O}(1)}$ algorithm under SETH (Theorem 1.4) (even for non-negative matrix A) no $f(d)(d+1)^{(1-\epsilon)\text{pw}}(mn)^{\mathcal{O}(1)}$ algorithm under SETH (Theorem 1.5) (even for non-negative matrix A)

Figure 1: A summary of our lower bound results in comparison to the known/new upper bound results for non-negative A . Here, n and m are the number of variables and constraints respectively, pw denotes the path-width of the column matroid of A and d denotes a bound on the largest entry in b .

In fact, they also show that the assumption of non-negativity is unavoidable (without any further assumptions such as a bounded domain for the variables) in this setting because (IP) is NP-hard when the constraint matrix A is allowed to have negative values (in fact even when restricted to $\{-1, 0, 1\}$) and the branchwidth of the column matroid of A is at most 3. A close inspection of the instances they construct in their NP-hardness reduction shows that the column matroids of the resulting constraint matrices are in fact direct sums of circuits, implying that even their *path-width* is bounded by 3. The parameter path-width is closely related to the notion of *trellis-width* of a linear code, which is a parameter commonly used in coding theory [20]. For a matrix $A \in \mathbb{R}^{m \times n}$, computing the path-width of the column matroid of A is equivalent to computing the trellis-width of the linear code generated by A . Roughly speaking, the path-width of the column matroid of A is at most k , if there is a permutation of the columns of A such that in the matrix A' obtained from A by applying this column-permutation, for every $1 \leq i \leq n-1$, the *dimension* of the subspace of \mathbb{R}^m obtained by taking the intersection of the subspace of \mathbb{R}^m spanned by the first i columns and the subspace of \mathbb{R}^m spanned by the remaining columns, is at most $k-1$.

The value of the parameter path-width is always at least the value of branch-width and at most $\text{rank}+1$. As a result, any upper bounds on the complexity of (IP) in terms of path-width will translate to upper bounds in terms of the larger parameter rank (number of constraints) and any lower bounds on the complexity of (IP) in terms of path-width will translate to lower bounds in terms of the smaller parameter branch-width. Motivated by this fact, we study the question of designing an ‘optimal’ pseudo-polynomial time algorithm for (IP) when the column matroid of A has constant path-width. We first obtain the following upper bound.

Theorem 1.3. *(IP) with non-negative $m \times n$ matrix A given together with a path decomposition of its column matroid of width k is solvable in time $\mathcal{O}((d+1)^{k+1}mn + m^2n)$, where $d = \max\{b_1, \dots, b_m\}$.*

As mentioned earlier, the NP-hardness of (IP) on constant branch-width instances also holds for constant path-width instances and hence the assumption of non-negativity is unavoidable here as well. Furthermore, while the proof of this theorem is not hard and is in fact almost identical to the proof of Theorem 1.2, this upper bound becomes really interesting when placed in context and compared to the tight lower bounds we provide in our next two theorems, which form the main technical part of the paper. In these theorems, we provide tight conditional (subject to Strong ETH) lower bounds for (IP) matching the running time of the algorithm of Theorem 1.3 (see Figure 1). Strong ETH (SETH) is the conjecture that CNF-SAT cannot be solved in time $(2-\epsilon)^n m^{\mathcal{O}(1)}$ on n -variable m -clause formulas for any constant ϵ . Both ETH and SETH were first introduced in

the work of Impagliazzo and Paturi [21], which built upon earlier work of Impagliazzo, Paturi and Zane [22]. We obtain the following lower bounds for (IP). The first result shows that we cannot relax the $(d+1)^k$ factor in Theorem 1.3 *even* if we allow in the running time, an arbitrary function depending on k . The second result shows a similar lower bound in terms of d instead of k . Put together the results imply that no matter how much one is allowed to compromise on either the path-width or the bound on d , it is unlikely that the algorithm of Theorem 1.3 can be improved.

Theorem 1.4. *Unless SETH fails, (IP) with even a non-negative $m \times n$ constraint matrix A cannot be solved in time $f(k)(d+1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any function f and $\epsilon > 0$, where $d = \max\{b_1, \dots, b_m\}$ and k is the path-width of the column matroid of A .*

Theorem 1.5. *Unless SETH fails, (IP) with even a non-negative $m \times n$ constraint matrix A cannot be solved in time $f(d)(d+1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any function f and $\epsilon > 0$, where $d = \max\{b_1, \dots, b_m\}$ and k is the path-width of the column matroid of A .*

Although the proofs of both lower bounds have a similar *structure*, we believe that there are sufficiently many differences in the proofs to warrant stating them separately. Finally, since the branch-width of a matroid never exceeds its path-width, our lower bounds hold when the parameter of interest is chosen to be the branch-width of the column matroid of A as well. That is, under SETH, there is no $f(\text{bw})(d+1)^{(1-\epsilon)\text{bw}}(mn)^{\mathcal{O}(1)}$ or $f(d)(d+1)^{(1-\epsilon)\text{bw}}(mn)^{\mathcal{O}(1)}$ algorithm for (IP) with non-negative constraint matrices (where bw denotes the branchwidth of the column matroid of A), almost matching the upper bound of $\mathcal{O}((d+1)^{2\text{bw}}mn + m^2n)$ from Theorem 1.4.

Related work. Currently ETH is a commonly accepted conjecture and it serves as the basic tool used for establishing asymptotically optimal lower bounds for various parameterized and exact exponential algorithms. While there is no such consensus on SETH, the hypothesis has already played a crucial role in the recent spectacular and rapid development in the analyses of polynomial, parameterized and exact exponential algorithms. In particular, SETH was used to establish conditional tight lower bounds for a number of fundamental computational problems, including k -domination [27], the diameter of sparse graphs [29], dynamic connectivity problems [2], the Frechet distance computation [5], string editing distance [3], dynamic programming on graphs of bounded tree-width and clique-width [24, 12, 15, 9], Steiner tree and subset sum [10], finding the longest common subsequence and the dynamic time warping distance [1, 6], and matching regular expressions [4]. For further overview of applications of ETH and SETH, we refer to surveys [25, 31] as well as [11, Chapter 14]. Our work extends this line of research by adding the fundamental (IP) problem to the classes of “SETH-hard” and “ETH-hard” problems.

Organization of the paper. The remaining part of the paper is organized as follows. The main technical part of the paper is devoted to proving Theorem 1.4 and Theorem 1.5. Therefore, once we have set up the requisite preliminary definitions, we begin with Section 3 where we prove Theorem 1.4. The first part of this section contains an overview of both reductions and could be helpful for the reader in navigating the paper. We then prove Theorem 1.5 in Section 4 and Theorem 1.3 in Section 5 (completing the results for constant path-width) and that of Theorem 1.1 in Section 6.

2 Preliminaries

We assume that the reader is familiar with basic definitions from linear algebra, matroid theory and graph theory.

Notations. We use $\mathbb{Z}_{\geq 0}$ and \mathbb{R} to denote the set of non negative integers and real numbers, respectively. For any positive integer n , we use $[n]$ and \mathbb{Z}_n to denote the sets $\{1, \dots, n\}$ and $\{0, 1, \dots, n-1\}$, respectively. For convenience, we say that $[0] = \emptyset$. For any two vectors $b, b' \in \mathbb{R}^m$ and $i \in [m]$, we use $b[i]$ to denote the i^{th} coordinate of b and we write $b' \leq b$, if $b'[i] \leq b[i]$ for all $i \in [m]$. We often use 0 to denote the zero-vector whose length will be clear from the context. For a matrix $A \in \mathbb{R}^{m \times n}$, $I \subseteq [m]$ and $J \subseteq [n]$, $A[I, J]$ denote the submatrix of A obtained by the restriction of A to the rows indexed by I and columns indexed by J . For an $m \times n$ matrix A and n -vector v , we can write $Av = \sum_{i=1}^n A_i v[i]$, where A_i is the i^{th} column of A . Here we say that $v[i]$ is a multiplier of column A_i .

Branch-width of matroids. The notion of the branch-width of graphs, and implicitly of matroids, was introduced by Robertson and Seymour in [28]. Let $M = (U, \mathcal{F})$ be a matroid with universe set U and family \mathcal{F} of independent sets over U . We use r_M to denote the rank function of M . That is, for any $S \subseteq U$, $r_M(S) = \max_{S' \subseteq S, S' \in \mathcal{F}} |S'|$. For $X \subseteq U$, the *connectivity function* of M is defined as

$$\lambda_M(X) = r_M(X) + r_M(U \setminus X) - r_M(U) + 1$$

For matrix $A \in \mathbb{R}^{m \times n}$, we use $M(A)$ to denote the column-matroid of A . In this case the connectivity function $\lambda_{M(A)}$ has the following interpretation. For $E = \{1, \dots, n\}$ and $X \subseteq E$, we define

$$S(A, X) = \text{span}(A|X) \cap \text{span}(A|E \setminus X),$$

where $A|X$ is the set of columns of A restricted to X and $\text{span}(A|X)$ is the subspace of \mathbb{R}^m spanned by the columns $A|X$. It is easy to see that the dimension of $S(A, X)$ is equal to $\lambda_{M(A)}(X) - 1$.

A tree is *cubic* if its internal vertices all have degree 3. A *branch decomposition* of matroid M with universe set U is a cubic tree T and mapping μ which maps elements of U to leaves of T . Let e be an edge of T . Then the forest $T - e$ consists of two connected components T_1 and T_2 . Thus every edge e of T corresponds to the partitioning of U into two sets X_e and $U \setminus X_e$ such that $\mu(X_e)$ are the leaves of T_1 and $\mu(U \setminus X_e)$ are the leaves of T_2 . The *width* of edge e is $\lambda_M(X_e)$ and the width of branch decomposition (T, μ) is the maximum edge width, where maximum is taken over all edges of T . Finally, the *branch-width* of M is the minimum width taken over all possible branch decompositions of M .

The *path-width* of a matroid is defined as follows. Let us remind that a *caterpillar* is a tree which is obtained from a path by attaching to some vertices of the paths some leaves. Then the path-width of a matroid is the minimum width of a branch decomposition (T, μ) , where T is a cubic caterpillar. Let us note that every mapping of elements of a matroid to the leaves of a cubic caterpillar, correspond to their ordering. Jeong, Kim, and Oum [23] gave a constructive fixed-parameter tractable algorithm to construct a path decomposition of width at most k for a column matroid of a given matrix.

ETH and SETH. For $q \geq 3$, let δ_q be the infimum of the set of constants c for which there exists an algorithm solving q -SAT with n variables and m clauses in time $2^{cn} \cdot m^{\mathcal{O}(1)}$. The *Exponential-Time Hypothesis (ETH)* and *Strong Exponential-Time Hypothesis (SETH)* are then formally defined as follows. ETH conjectures that $\delta_3 > 0$ and SETH that $\lim_{q \rightarrow \infty} \delta_q = 1$.

3 Proof of Theorem 1.4

In this section we prove that unless SETH fails, (IP) with non-negative matrix A cannot be solved in time $f(k)(d+1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any function f and $\epsilon > 0$, where $d = \max\{b[1], \dots, b[m]\}$ and k is the path-width of the column matroid of A . In Subsection 3.1, we give an overview of our reductions and in Subsection 3.2 we give a detailed proof of Theorem 1.4

3.1 Overview of our reductions

We prove Theorems 1.4 and 1.5 by giving reductions from CNF-SAT, where the parameters in the reduced instances are required to obey certain strict conditions. For example, the reduction we give to prove Theorem 1.4 must output an instance of (IP), where the path-width of the column matroid $M(A)$ of the constraint matrix A is a constant. Similarly, in the reduction used to prove Theorem 1.5, we need to construct an instance of (IP), where the largest entry in the target vector is upper bounded by a constant. These stringent requirements on the parameters make the SETH-based reductions quite challenging. However, reductions under SETH can take super polynomial time—they can even take $2^{(1-\epsilon)n}$ time for some $\epsilon > 0$, where n is the number of variables in the instance of CNF-SAT. This freedom to avail exponential time in SETH-based reductions is used crucially in the proofs of Theorems 1.4 and 1.5.

Now we give an overview of the reduction used to prove Theorem 1.4. Let ψ be an instance of CNF-SAT with n variables and m clauses. Given ψ and a fixed constant $c \geq 2$, we construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ of (IP) satisfying certain properties. Since for every $c \geq 2$, we have a different $A_{(\psi,c)}$ and $b_{(\psi,c)}$, this can be viewed as a family of instances of (IP). In particular our main technical lemma is the following.

Lemma 3.1. *Let ψ be an instance of CNF-SAT with n variables and m clauses. Let $c \geq 2$ be a fixed integer. Then, in time $\mathcal{O}(m^2 2^{\frac{n}{c}})$, we can construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP) with the following properties.*

- (a.) ψ is satisfiable if and only if $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ is feasible.
- (b.) The matrix $A_{(\psi,c)}$ is non-negative and has dimension $\mathcal{O}(m) \times \mathcal{O}(m 2^{\frac{n}{c}})$.
- (c.) The path-width of the column matroid of $A_{(\psi,c)}$ is at most $c + 4$.
- (d.) The largest entry in $b_{(\psi,c)}$ is at most $2^{\lceil \frac{n}{c} \rceil} - 1$.

Once we have Lemma 3.1, the proof of Theorem 1.4 follows from the following observation: if we have an algorithm \mathcal{A} solving (IP) in time $f(k)(d+1)^{(1-\epsilon)k}(mn)^a$ for some $\epsilon, a > 0$, then we can use this algorithm to refute SETH. In particular, given an instance ψ of CNF-SAT, we choose an appropriate c depending only on ϵ and a , construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP), and run \mathcal{A} on it. Our careful choice of c will imply a faster algorithm for CNF-SAT, refuting SETH. More formally, we choose c to be an integer such that $(1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c} < 1$. Then the total running time to test whether ψ is satisfiable, is the time required to construct $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ plus the time required by \mathcal{A} to solve the constructed instance of (IP). That is, the time required to test whether ψ is satisfiable is

$$\mathcal{O}(m^2 2^{\frac{n}{c}}) + f(c+4) 2^{\frac{n}{c}(1-\epsilon)(c+4)} 2^{\frac{a-n}{c}} m^{\mathcal{O}(1)} = 2^{((1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c})n} m^{\mathcal{O}(1)} = 2^{\epsilon' n} m^{\mathcal{O}(1)},$$

where $\epsilon' < 1$ is a constant depending on the choice of c . It is important to note that the utility of the reduction described in Lemma 3.1 is extremely sensitive to the value of the numerical parameters involved. In particular, even when the path-width blows up slightly, say up to δc , or when the largest entry in $b_{(\psi,c)}$ blows up slightly, say up to $2^{\delta \frac{n}{c}}$, for some $\delta > 1$, then the calculation above will *not* give us the desired refutation of SETH. Thus, the challenging part of the reduction described in Lemma 3.1 is making it work under these strict restrictions on the relevant parameters.

As stated in Lemma 3.1, in our reduction, we need to obtain a constraint matrix with small path-width. An important first step towards this is understanding what a matrix of small path-width looks like. We first give an intuitive description of the structure of such matrices. Let A be a $m \times n$ matrix of small path-width and let $M(A)$ be the column matroid of A . For any $i \in \{1, \dots, n-1\}$,

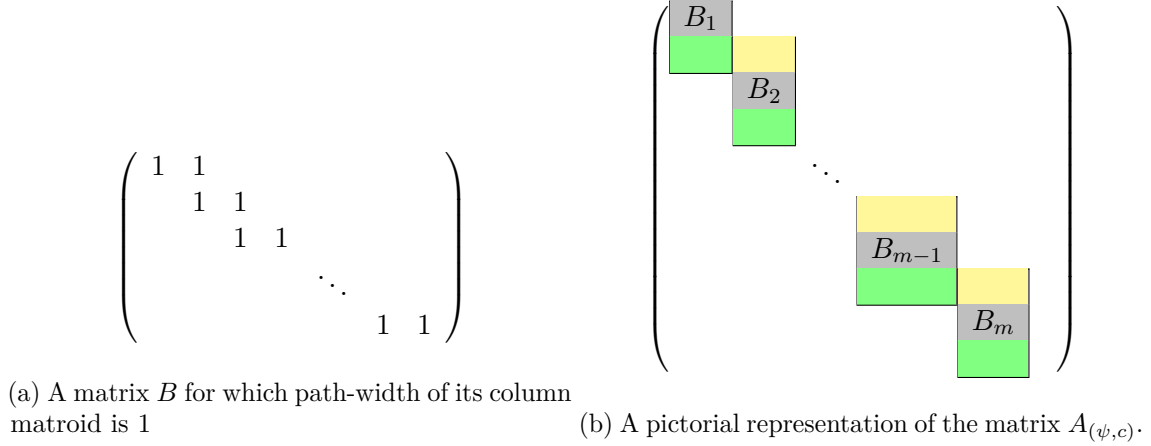


Figure 2: Comparison of $A_{(\psi,c)}$ with a low path-width matrix.

let $A|\{1, \dots, i\}$ denote the set of columns (or vectors) in A whose index is at most i (that is, the first i columns) and let $A|\{i+1, \dots, n\}$ denote the set of columns with index strictly greater than i . The path-width of $M(A)$ is at most

$$\max_i \dim(\text{span}(A|\{1, \dots, i\}) \cap \text{span}(A|\{i+1, \dots, n\})) + 1.$$

Hence, in order to obtain a bound on the pathwidth, it is sufficient to bound $\dim(\text{span}(A|\{1, \dots, i\}) \cap \text{span}(A|\{i+1, \dots, n\}))$ for every $i \in [n]$. Consider for example, the matrix B given in Figure 2a. The path-width of $M(B)$ is clearly at most 1. In *our* reduced instance, the constructed constraint matrix $A_{(\psi,c)}$ will be an appropriate extension of B . That is $A_{(\psi,c)}$ will have the “same form” as B but with each 1 replaced by a submatrix of order $\mathcal{O}(c) \times n'$ for some n' . See Fig. 2b for a pictorial representation of $A_{(\psi,c)}$.

The construction used in Lemma 3.1 takes as input an instance ψ of CNF-SAT with n variables and a fixed integer $c \geq 2$, and outputs an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP), that satisfies all four properties of the lemma. Let X denote the set of variables in the input CNF-formula $\psi = C_1 \wedge C_2 \wedge \dots \wedge C_m$. For the purposes of the present discussion we assume that c divides n . We partition the variable set X into c blocks X_0, \dots, X_{c-1} , each of size $\frac{n}{c}$. Let \mathcal{X}_i , $i \in \{0, \dots, c-1\}$, denote the set of assignments of variables corresponding to X_i . Set $\ell = \frac{n}{c}$ and $L = 2^\ell$. Clearly, the size of \mathcal{X}_i is upper bounded by $2^{\frac{n}{c}} = 2^\ell = L$. We denote the assignments in \mathcal{X}_i by $\phi_0(X_i), \phi_1(X_i), \dots, \phi_{L-1}(X_i)$. To construct the matrix $A_{(\psi,c)}$, we view “each of these assignments as a different assignment for each clause”. In other words we have separate sets of column vectors in the constraint matrix $A_{(\psi,c)}$ corresponding to different pairs (C_r, X_i) , where C_r is a clause and X_i is a block in the partition of X . All the values set in these columns are based on the assignments of X_i and the clause C_r . That is, based on the clause C_r and assignments in \mathcal{X}_i . In total we have $2L$ columns corresponding to (C_r, X_i) . The set of columns corresponding to C_r , that is, the set of columns corresponding to (C_r, X_i) , for all i , together forms a bigger block of columns, denoted by $A_{(\psi,c)}^{C_r}$, in $A_{(\psi,c)}$. The columns of $A_{(\psi,c)}^{C_r}$ appears consecutively in $A_{(\psi,c)}$. In other words, the clauses of ψ partition the set of columns of $A_{(\psi,c)}$ into $A_{(\psi,c)}^{C_r}$, $r \in \{1, \dots, m\}$, where columns in each of the parts $(A_{(\psi,c)}^{C_r})$ occur consecutively. Thus, we can *identify* each column in the matrix $A_{(\psi,c)}$ with a pair $(C_r, \phi_j(X_i))$, $i \in \{0, \dots, c-1\}$ and $j \in \{0, \dots, L-1\}$. For a pair $(C_r, \phi_j(X_i))$, we refer to $\phi_j(X_i)$ as the *assignment* part of the pair. The non-zero values in $A_{(\psi,c)}^{C_r}$ are covered by $4c+1$ specific consecutive rows. These $4c+1$ rows are divided into 3 parts according to their roles in the

reduction:

- the first $2c$ rows comprise the *predecessor matching part*,
- the middle row is called the *evaluation part*, and
- the $2c$ rows after the evaluation part comprise the *successor matching part*.

The entries in the row corresponding to the evaluation part get values 0 or 1 depending on whether the assignment part of the pair $(C_r, \phi_j(X_i))$ satisfies C_r or not.

The matrix $A_{(\psi,c)}$ and the target vector $b_{(\psi,c)}$ are constructed in such a way that all the feasible solutions to $A_{(\psi,c)}x = b_{(\psi,c)}$ are from the set $\{0, 1\}^{n'}$, where n' is the set of columns in $A_{(\psi,c)}$. Hence, setting a coordinate of x to 1 corresponds to *choosing* a particular column from $A_{(\psi,c)}$. In our reduction we use a *selector gadget* to enforce that any feasible solution will choose *exactly* one column from the set of columns corresponding to a pair (C_r, X_i) . That is, it corresponds to *choosing* a column identified by $(C_r, \phi_j(X_i))$. Thus it results in choosing an assignment $\phi_j(X_i)$ to the variables in the set X_i . Note that this implies that we will choose an assignment in \mathcal{X}_i for each clause C_r . That way we might choose m assignments from \mathcal{X}_i corresponding to m different clauses. However, for the backward direction of the proof, it is important that we choose the *same* assignment from \mathcal{X}_i for each clause. This will ensure that we have selected an assignment to the variables in X_i . Towards this we assign values in each of the columns in a way that all the assignments chosen by a feasible solution for a particular block across different clauses are the same. Then choosing two columns, one from the set of columns corresponding to (C_r, X_i) and the other from the columns of $(C_{r'}, X_i)$ in a feasible solution would imply that both of these columns correspond to one *particular* assignment of X_i . In this case, we say that these two columns are *consistent*. We enforce these consistencies in a *sequential* manner. That is, for any block X_i , we make sure that the two columns chosen among the columns corresponding to (C_r, X_i) and (C_{r+1}, X_i) are consistent for any $r \in \{1, \dots, m-1\}$, as opposed to checking the consistency for every pair (C_r, X_i) and $(C_{r'}, X_i)$ for $r \neq r'$. Thus in some sense these consistencies *propagate*. Such a propagation of consistencies is realized through rows corresponding to the predecessor matching part and the successor matching part. For that the rows corresponding to predecessor matching part of $A_{(\psi,c)}^{C_r}$ will be the *same* as the successor matching part of $A_{(\psi,c)}^{C_{r-1}}$ and the rows corresponding to the successor matching part of $A_{(\psi,c)}^{C_r}$ will be the *same* as the predecessor matching part of $A_{(\psi,c)}^{C_{r+1}}$. Both the predecessor matching part as well as the successor matching part contain *designated rows* for each block X_i of variables to handle consistencies between (C_r, X_i) and (C_{r+1}, X_i) . Recall that \mathcal{X}_i denotes the set of assignments of X_i and $|\mathcal{X}_i| = 2^\ell = L$. Furthermore, assignments in \mathcal{X}_i are denoted by $\phi_0(X_i), \dots, \phi_{L-1}(X_i)$. Thus, we can identify the assignment $\phi_j(X_i)$ by a non-negative integer $j \leq L-1$. These values are assigned in a co-ordinated manner at designated places in the predecessor matching part as well as in the successor matching part, enabling us to argue consistency. The largest entry in $b_{(\psi,c)}$ is upper bounded by $L-1$. Furthermore, the idea of making consistency in a sequential manner also allows us to bound the path-width of column matroid of $A_{(\psi,c)}$ by $c+4$.

The proof technique for Theorem 1.5 is similar to that for Theorem 1.4. This is achieved by modifying the matrix $A_{(\psi,c)}$ constructed in the reduction described for Lemma 3.1. The largest entry in $A_{(\psi,c)}$ is $2^{\frac{n}{c}} - 1$. So each of these values can be represented by a binary string of length at most $\ell = \frac{n}{c}$. We remove each row, say row indexed by γ , with entries greater than 1 and replace it with $\frac{n}{c}$ rows, $\gamma_1, \dots, \gamma_\ell$. Where, for any j , if the value $A_{(\psi,c)}[\gamma, j] = W$ then $A_{(\psi,c)}[\gamma_k, j] = \eta_k$, where η_k is the k^{th} bit in the ℓ -sized binary representation of W . This modification reduces the largest entry in $A_{(\psi,c)}$ to 1 and increases the path-width from constant to approximately n . Finally, we set all the entries in $b_{(\psi,c)}$ to be 1. This concludes the overview of our reductions and we now proceed to a detailed exposition.

3.2 Detailed Proof of Theorem 1.4

Towards the proof of Theorem 1.4, we first present the proof of our main technical lemma (Lemma 3.1), which we restate here for the sake of completeness.

Lemma 3.1. *Let ψ be an instance of CNF-SAT with n variables and m clauses. Let $c \geq 2$ be a fixed integer. Then, in time $\mathcal{O}(m^2 2^{\frac{n}{c}})$, we can construct an instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$, of (IP) with the following properties.*

- (a.) ψ is satisfiable if and only if $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ is feasible.
- (b.) The matrix $A_{(\psi,c)}$ is non-negative and has dimension $\mathcal{O}(m) \times \mathcal{O}(m 2^{\frac{n}{c}})$.
- (c.) The path-width of the column matroid of $A_{(\psi,c)}$ is at most $c + 4$.
- (d.) The largest entry in $b_{(\psi,c)}$ is at most $2^{\lceil \frac{n}{c} \rceil} - 1$.

Let $\psi = C_1 \wedge C_2 \wedge \dots \wedge C_m$ be an instance of CNF-SAT with variable set $X = \{x_1, x_2, \dots, x_n\}$ and let $c \geq 2$ be a fixed constant given in the statement of Lemma 3.1. We construct the instance $A_{(\psi,c)}x = b_{(\psi,c)}, x \geq 0$ of (IP) as follows.

Construction. Let $\mathcal{C} = \{C_1, \dots, C_m\}$. Without loss of generality, we assume that n is divisible by c , otherwise we add at most c dummy variables to X such that $|X|$ is divisible by c . We divide X into c blocks X_0, X_1, \dots, X_{c-1} . That is $X_i = \{x_{\frac{i \cdot n}{c} + 1}, x_{\frac{i \cdot n}{c} + 2}, \dots, x_{\frac{(i+1) \cdot n}{c}}\}$ for each $i \in \mathbb{Z}_c$. Let $\ell = \frac{n}{c}$ and $L = 2^\ell$. For each block X_i , there are exactly 2^ℓ assignments. We denote these assignments by $\phi_0(X_i), \phi_1(X_i), \dots, \phi_{L-1}(X_i)$.

Now we create m matrices, one for each clause $C \in \mathcal{C}$. These matrices will be submatrices of the constraint matrix $A_{(\psi,c)}$. For each clause $C_r \in \mathcal{C}$, we create a $(4c + 1) \times c \cdot 2^{\ell+1}$ matrix B_r . For each block X_i and all possible assignments to the variables of X_i , we allocate $2^{\ell+1}$ columns in B_r . For each assignment $\phi_j(X_i)$ there are two columns in B_r corresponding to it. Then the first $2^{\ell+1}$ columns of B_r correspond to assignments of X_0 , the second $2^{\ell+1}$ columns correspond to assignments of X_1 , etc.

Matrices B_r for $1 < r < m$. We first define B_r for indices $1 < r < m$. Matrices B_1 and B_m have a slightly different structure compared to the other matrices and so we define them separately. The non-zero values of B_r are defined as follows. Each assignment $\phi_j(X_i)$ is identified by the number j . Each $\phi_j(X_i)$ defines 8 entries in B_r : four in the column numbered $2^{\ell+1}i + 2j + 1$ and four in the column numbered $2^{\ell+1}i + 2j + 2$. The rows of B_r are partitioned into 3 parts. The part composed of the first $2c$ rows is called the *predecessor matching part*, the part composed of the row indexed by $2c + 1$ is called the *evaluation part*, and the part composed of the last $2c$ rows is called the *successor matching part*, see Fig. 3a.

The predecessor matching part is defined by

$$B_r[2i + 1, i \cdot 2^{\ell+1} + 2j + 1] = B_r[2i + 1, i \cdot 2^{\ell+1} + 2j + 2] = j, i \in \mathbb{Z}_c. \quad (1)$$

For $i \in \mathbb{Z}_c$, the evaluation part is defined by

$$B_r[2c + 1, i \cdot 2^{\ell+1} + 2j + 1] = 0, \quad (2)$$

and

$$B_r[2c + 1, i \cdot 2^{\ell+1} + 2j + 2] = \begin{cases} 1, & \text{if } \phi_j(X_i) \text{ satisfies } C_r, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

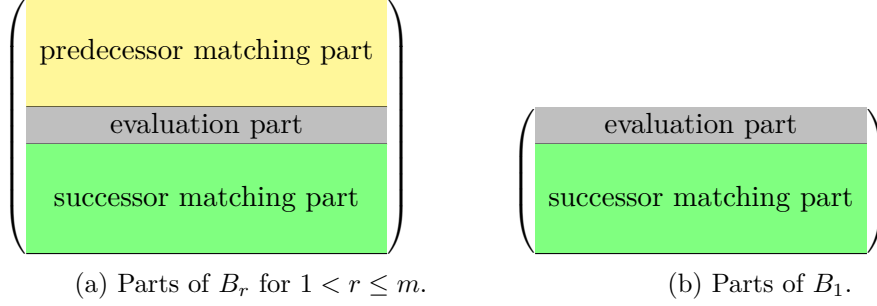


Figure 3: Parts of B_m .

The successor matching part for B_r is defined for $i \in \mathbb{Z}_c$ as

$$B_r[2c + 2i + 2, i \cdot 2^{\ell+1} + 2j + 1] = B_r[2c + 2i + 2, i \cdot 2^{\ell+1} + 2j + 2] = L - 1 - j, \quad (4)$$

$$B_r[2c + 2i + 3, i \cdot 2^{\ell+1} + 2j + 1] = B_r[2c + 2i + 3, i \cdot 2^{\ell+1} + 2j + 2] = 1. \quad (5)$$

All other entries in B_r , which are not defined above, are set to zero. That is, for all $i, i' \in \mathbb{Z}_c$ and $a \in [2^{\ell+1}]$ such that $i \neq i'$,

$$B_r[2i + 1, i' \cdot 2^{\ell+1} + a] = B_r[2i + 2, i' \cdot 2^{\ell+1} + a] = 0, \quad (6)$$

$$B_r[2i + 2, i \cdot 2^{\ell+1} + a] = 0, \text{ and} \quad (7)$$

$$B_r[2c + 2i + 2, i' \cdot 2^{\ell+1} + a] = B_r[2c + 2i + 3, i' \cdot 2^{\ell+1} + a] = 0. \quad (8)$$

Before describing the construction of B_1 and B_m , we provide a brief description of certain desirable properties possessed by B_r . We have designated set of columns per pair (C_r, X_i) , which are indexed by $[(i + 1)2^{\ell+1}] \setminus [i \cdot 2^{\ell+1}]$ and (5) ensures that at most one of the columns from this set is chosen by a feasible solution. This will be forced by putting 1 in the corresponding coordinate of vector $b_{(\psi, c)}$. In the construction of $A_{(\psi, c)}$, we will only add zeros to the entries in the row of $A_{(\psi, c)}$ corresponding to the $(2c + 2i + 3)^{th}$ row of B_r , but outside the submatrix B_r of $A_{(\psi, c)}$. This guarantees that exactly one of them is chosen by a feasible solution. The purpose of (1) is to ensure consistency with the column selected from (C_{r-1}, X_i) , and purpose of (4) is to ensure consistency with the column selected from (C_{r+1}, X_i) . We construct the matrix $A_{(\psi, c)}$ in such a way that the row of B_r indexed by $2i + 1$ and the row of B_{r-1} indexed by $2c + 2i + 2$ are equal in $A_{(\psi, c)}$. Suppose that this row is indexed by h in $A_{(\psi, c)}$. Then (1) and (4) ensure that if we choose consistent columns from the columns of (C_{r-1}, X_i) and (C_r, X_i) , then the sum of the values in coordinate h of the selected columns will be equal to $L - 1$. So we will set $b_{(\psi, c)}[h] = L - 1$ in the target vector $b_{(\psi, c)}$. For each assignment $\phi_j(X_i)$, we have two designated columns in B_r , they are indexed by $i \cdot 2^{\ell+1} + 2j + 1$ and $i \cdot 2^{\ell+1} + 2j + 2$. The reason for creating two columns instead of just one per $\phi_j(X_i)$ is the following. The coordinate j' of the target vector $b_{(\psi, c)}$ corresponding to the row which contains the row of B_r indexed by $2c + 1$ will be set to 1. For any satisfying assignment of ϕ of ψ , more than one partial assignments of ϕ (i.e, assignments of ϕ restricted to different blocks of X) may satisfy the clause C_r . So among the pairs of columns corresponding to these satisfying partial assignments a feasible solution will choose the first column from the pair for all but one. For a partial assignment (assignment of a block of X) which satisfies C_r , the feasible solution will choose the second column corresponding to it. Equations (2) and (3) make sure that the entries corresponding to the coordinate j' from the set of chosen columns by a feasible solution will add up to 1; hence at least one selected column would correspond to an assignment (of a block of X) satisfying clause C_r .

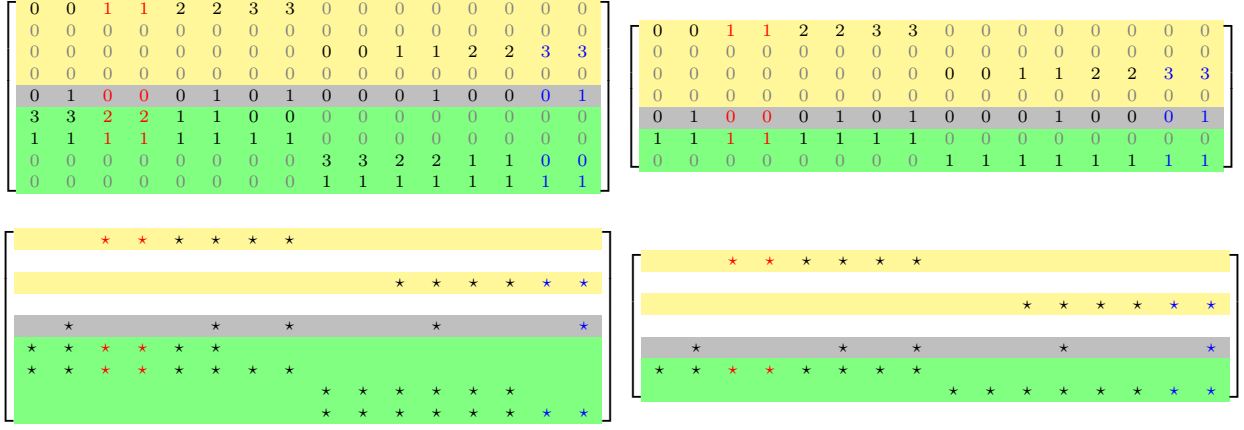


Figure 4: Let $n = 4, c = 2, \ell = 2$ and $C_r = x_1 \vee \overline{x_2} \vee x_4$. The assignments are $\phi_0(X_0) = \{x_1 = x_2 = 0\}$, $\phi_1(X_0) = \{x_1 = 0, x_2 = 1\}$, $\phi_2(X_0) = \{x_1 = 1, x_2 = 0\}$, $\phi_3(X_0) = \{x_1 = x_2 = 1\}$, $\phi_0(X_1) = \{x_3 = x_4 = 0\}$, $\phi_1(X_1) = \{x_3 = 0, x_4 = 1\}$, $\phi_2(X_1) = \{x_3 = 1, x_4 = 0\}$, $\phi_3(X_1) = \{x_3 = x_4 = 1\}$. The entries defined according to $\phi_1(X_0)$ and $\phi_3(X_1)$ are colored red and blue respectively. If $1 < r < m$, then the matrix on the left represents B_r and if $r = 1$, then B_r can be obtained by deleting the yellow colored portion from the left matrix. The matrix on the right represents B_m . Sometimes, it is helpful to focus on the positions containing non-zero elements. This can be found in the two matrices at the bottom of the figure.

Matrices B_1 and B_m . The matrix B_1 is created as above but with the exception that we remove the predecessor matching part (see Fig. 3b). The matrix B_m is created as above with the exception that we remove the rows numbered $2c + 2, 2c + 4, \dots, 4c$. An illustration of B_m is given in Fig. 4. Formally, the entries of B_1 and B_m , defined by $\phi_j(X_i)$ are given below.

For B_1 we define its entries as

$$B_1[2i + 2, i \cdot 2^{\ell+1} + 2j + 1] = B_1[2i + 2, i \cdot 2^{\ell+1} + 2j + 2] = L - j - 1, \quad (9)$$

$$B_1[2i + 3, i \cdot 2^{\ell+1} + 2j + 1] = B_1[2i + 3, i \cdot 2^{\ell+1} + 2j + 2] = 1, \quad (10)$$

$$B_1[1, i \cdot 2^{\ell+1} + 2j + 1] = 0, \text{ and} \quad (11)$$

$$B_1[1, i \cdot 2^{\ell+1} + 2j + 2] = \begin{cases} 1 & \text{if } \phi_j(X_i) \text{ satisfies } C_1, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

For B_m ,

$$B_m[2i + 1, i \cdot 2^{\ell+1} + 2j + 1] = B_m[2i + 1, i \cdot 2^{\ell+1} + 2j + 2] = j, \quad (13)$$

$$B_m[2c + i + 2, i \cdot 2^{\ell+1} + 2j + 1] = B_m[2c + i + 2, i \cdot 2^{\ell+1} + 2j + 2] = 1,$$

$$B_m[2c + 1, i \cdot 2^{\ell+1} + 2j + 1] = 0, \text{ and}$$

$$B_m[2c + 1, i \cdot 2^{\ell+1} + 2j + 2] = \begin{cases} 1, & \text{if } \phi_j(X_i) \text{ satisfies } C_m, \\ 0, & \text{otherwise.} \end{cases}$$

All other entries in B_1 and B_m , which are not defined above, are set to zero. That is, for all

$i, i' \in \mathbb{Z}_c$ and $a \in [2^{\ell+1}]$ such that $i \neq i'$,

$$B_1[2i + 2, i' \cdot 2^{\ell+1} + a] = B_1[2i + 3, i' \cdot 2^{\ell+1} + a] = 0, \quad (14)$$

$$B_m[2i + 1, i' \cdot 2^{\ell+1} + a] = B_m[2i + 2, i' \cdot 2^{\ell+1} + a] = 0, \quad (15)$$

$$B_m[2i + 2, i \cdot 2^{\ell+1} + a] = 0, \quad (16)$$

$$B_m[2c + i + 2, i' \cdot 2^{\ell+1} + a] = 0. \quad (17)$$

Matrix $A_{(\psi, c)}$ and vector $b_{(\psi, c)}$. Now we explain how to construct the constraint matrix $A_{(\psi, c)}$ and vector $b_{(\psi, c)}$, which would serve as instance of (IP). In what follows,

we simplify the notation by using A instead of $A_{(\psi, c)}$ and b instead of $b_{(\psi, c)}$.

The matrices B_1, \dots, B_m are disjoint submatrices of A and they cover all non zero entries of A . Informally, the submatrices B_1, \dots, B_m form a chain such that the rows corresponding to the successor matching part of B_r will be the same as the rows in the predecessor matching part of B_{r+1} . A pictorial representation of A can be found in Fig. 2b. Formally, A is a $((m-1)(2c+1) + (c+1)) \times (m \cdot c \cdot 2^{\ell+1})$ matrix. Let $I_1 = [2c+1]$ and $I_m = [(m-1)(2c+1) + (c+1)] \setminus [(m-1)(2c+1) - 2c]$. For every $1 < r < m$, let $I_r = [r(2c+1)] \setminus [(r-1)(2c+1) - 2c]$, and for $r \in [m]$, let $J_r = [r \cdot c \cdot 2^{\ell+1}] \setminus [(r-1) \cdot c \cdot 2^{\ell+1}]$. Now for each $r \in [m]$, we put matrix $A[I_r, J_r] := B_r$. All other entries of A not belonging to any of the submatrices $A[I_r, J_r]$ are set to zero. This completes the construction of A .

Now we define the $((m-1)(2c+1) + (c+1))$ -dimensional vector b . Let

$$P = \{(r-1)(2c+1) + 2j + 2 \mid r \in [m-1], j \in \mathbb{Z}_c\}.$$

In other words, P contains the indices of some rows: for each $r \in [m-1]$, alternating rows in the successor matching part (and thus the alternating rows in the predecessor matching part) of $A[I_r, J_r]$ belong to P (refer to Fig. 2b again). Then the entries of b are defined as

$$b[q] = \begin{cases} L-1, & \text{if } q \in P, \\ 1, & \text{otherwise.} \end{cases}$$

This completes the construction of the matrix A and vector b which together make up the required instance of (IP).

Correctness. Now we prove that ψ is satisfiable if and only if there is a non-negative integer vector x such that $Ax = b$. We start with some notations. We partition the set of columns of A into m parts J_1, \dots, J_m (we have already defined these sets) with one part per clause. For each $r \in [m]$, J_r is the set of columns associated with C_r . We further divide J_r into c equal parts, one per variable set X_i . These parts are

$$P_{r,i} = \{(r-1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 1, \dots, (r-1)c \cdot 2^{\ell+1} + (i+1) \cdot 2^{\ell+1}\}, i \in \mathbb{Z}_c.$$

In other words, $P_{r,i}$ is the set of columns associated with the tuple (C_r, X_i) and $|P_{r,i}| = 2^{\ell+1}$. The set $P_{r,i}$ is divided into 2^ℓ parts of size two each, one per tuple $(C_r, \phi_j(X_i))$, where $C_r \in \mathcal{C}, j \in \mathbb{Z}_L$ and $i \in \mathbb{Z}_c$. The two columns associated with tuple $(C_r, \phi_j(X_i))$ are indexed by $(r-1)2^{\ell+1} + i \cdot 2^{\ell+1} + 2j + 1$ and $(r-1)2^{\ell+1} + i \cdot 2^{\ell+1} + 2j + 2$ in A . We also put $n' = m \cdot c \cdot 2^{\ell+1}$ to be the number of columns in A .

Lemma 3.1. *Formula ψ is satisfiable if and only if there exists $x^* \in \mathbb{Z}_{\geq 0}^{n'}$ such that $Ax^* = b$.*

Proof. Suppose ψ is satisfiable and let ϕ be its satisfying assignment. There exists $j_0, j_1, \dots, j_{c-1} \in \mathbb{Z}_L$ such that ϕ is the union of $\phi_{j_0}(X_0), \phi_{j_1}(X_1), \dots, \phi_{j_{c-1}}(X_{c-1})$. Each clause $C \in \mathcal{C}$, C is satisfied by at least one of the assignments $\phi_{j_0}(X_0), \phi_{j_1}(X_1), \dots, \phi_{j_{c-1}}(X_{c-1})$. For each C , we fix an arbitrary $i \in \mathbb{Z}_c$ such that the assignment $\phi_{j_i}(X_i)$ satisfies clause C . Let α be a function which fixes these assignments for each clause. That is, $\alpha : \mathcal{C} \rightarrow \mathbb{Z}_c$ such that the assignment $\phi_{\alpha(C)}(X_{\alpha(C)})$ satisfies the clause C for every $C \in \mathcal{C}$. Now we define $x^* \in \mathbb{Z}_{\geq 0}^{n'}$ and prove that $Ax^* = b$. Let

$$\begin{aligned} Q_1 &= \{(r-1)2^{\ell+1} + i \cdot 2^{\ell+1} + 2j_i + 1 \mid r \in [m], i \in \mathbb{Z}_c, \alpha(C_r) \neq i\}, \\ Q_2 &= \{(r-1)2^{\ell+1} + i \cdot 2^{\ell+1} + 2j_i + 2 \mid r \in [m], i \in \mathbb{Z}_c, \alpha(C_r) = i\}, \text{ and} \\ Q &= Q_1 \cup Q_2. \end{aligned}$$

Then the vector x^* is defined by setting

$$x^*[q] = \begin{cases} 1, & \text{if } q \in Q, \\ 0, & \text{otherwise.} \end{cases}$$

The q -th entry in x^* is the multiplier of column q and so we say that entry $x^*[q]$ *corresponds* to column q . For each tuple $(C, \phi_{j_i}(X_i))$, one entry of x^* among the two entries corresponding to the columns associated with $(C, \phi_{j_i}(X_i))$ is set to 1. If $\alpha(C) = i$, then the second column corresponding to $(C, \phi_{j_i}(X_i))$ is set to 1, otherwise the first column corresponding to $(C, \phi_{j_i}(X_i))$ is set to 1. All other entries are set to zero. Also note that for every $r \in [m]$ and $i \in \mathbb{Z}_c$, we have that $|P_{r,i} \cap Q| = 1$ and let $\{q_{r,i}\} = P_{r,i} \cap Q$. Here notice that among the $2^{\ell+1}$ columns of $P_{r,i}$, exactly one column, which is indexed by $q_{r,i}$, belongs to Q . The column $q_{r,i}$ corresponds to one of the two columns corresponding to $(C_r, \phi_{j_i}(X_i))$.

We need the following auxiliary claims.

Claim 3.2. *For every $r \in [m-1]$ and $i, i' \in \mathbb{Z}_c$ such that $i \neq i'$, we have*

- (a) $\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i + 2, z] \cdot x[z] = L - 1 - j_i,$
- (b) $\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i + 3, z] \cdot x[z] = 1,$
- (c) $\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i' + h + 1, z] \cdot x[z] = 0, \text{ for any } h \in \{1, 2\}.$

Proof. First consider the case when $r = 1$. Let $P_{r,i} \cap Q = \{i \cdot 2^{\ell+1} + 2j_i + g\}$, where $g \in \{1, 2\}$. Then

$$\begin{aligned} \sum_{z \in P_{1,i}} A[2i + 2, z] \cdot x[z] &= \sum_{z \in P_{1,i} \cap Q} B_1[2i + 2, z] \\ &= B_1[2i + 2, i \cdot 2^{\ell+1} + 2j_i + g] \\ &= L - j_i - 1 \end{aligned} \tag{by (9)}$$

and (a) follows.

To prove (b), we have

$$\begin{aligned} \sum_{z \in P_{1,i}} A[2i + 3, z] \cdot x[z] &= \sum_{z \in P_{1,i} \cap Q} B_1[2i + 3, z] \\ &= B_1[2i + 3, i \cdot 2^{\ell+1} + 2j_i + g] \\ &= 1. \end{aligned} \tag{By (10)}$$

To show (c), observe that for $h \in \{1, 2\}$,

$$\begin{aligned}
\sum_{z \in P_{r,i}} A[2i' + h + 1, z] \cdot x[z] &= \sum_{z \in P_{r,i} \cap Q} B_1[2i' + h + 1, z] \\
&= B_1[2i' + h + 1, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= 0. \tag{By (14)}
\end{aligned}$$

Now consider the case when $r > 1$. Let $P_{r,i} \cap Q = \{(r-1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 2j_i + g\}$, where $g \in \{1, 2\}$. For this case we have

$$\begin{aligned}
\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i + 2, z] \cdot x[z] &= \sum_{z \in P_{r,i} \cap Q} B_r[2c + 2i + 2, z - ((r-1)c \cdot 2^{\ell+1})] \\
&= B_r[2c + 2i + 2, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= L - j_i - 1. \tag{By (4)}
\end{aligned}$$

$$\begin{aligned}
\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i + 3, z] \cdot x[z] &= B_r[2c + 2i + 3, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= 1. \tag{By (5)}
\end{aligned}$$

Finally, for $h \in \{1, 2\}$,

$$\begin{aligned}
\sum_{z \in P_{r,i}} A[(r-1)(2c+1) + 2i' + h + 1, z] \cdot x[z] &= \sum_{z \in P_{r,i} \cap Q} B_r[2c + 2i' + h + 1, z - ((r-1)c \cdot 2^{\ell+1})] \\
&= B_r[2c + 2i' + h + 1, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= 0. \tag{By (8)}
\end{aligned}$$

□

Claim 3.3. For every $r \in [m-1]$ and $i, i' \in \mathbb{Z}_c$ such that $i \neq i'$, we have

- (a) $\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i + 2, z] \cdot x[z] = j_i$,
- (b) $\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i + 3, z] \cdot x[z] = 0$,
- (c) $\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i' + h + 1, z] \cdot x[z] = 0$, where $h \in \{1, 2\}$.

Proof. The proof of the claim is similar to the proof of Claim 3.2. Let $P_{r+1,i} \cap Q = \{rc \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 2j_i + g\}$ where $g \in \{1, 2\}$. Then

$$\begin{aligned}
\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i + 2, z] \cdot x[z] &= B_{r+1}[2i + 1, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= j_i \tag{By (1)}
\end{aligned}$$

$$\begin{aligned}
\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i + 3, z] \cdot x[z] &= \sum_{z \in P_{r+1,i} \cap Q} B_{r+1}[2i + 2, z - (rc \cdot 2^{\ell+1})] \\
&= B_{r+1}[2i + 2, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= 0 \tag{By (7) or (16)}
\end{aligned}$$

For any $h \in \{1, 2\}$,

$$\begin{aligned}
\sum_{z \in P_{r+1,i}} A[(r-1)(2c+1) + 2i' + h + 1, z] \cdot x[z] &= \sum_{z \in P_{r+1,i} \cap Q} B_{r+1}[2i' + h, z - (rc \cdot 2^{\ell+1})] \\
&= B_{r+1}[2i' + h, i \cdot 2^{\ell+1} + 2j_i + g] \\
&= 0 \quad (\text{By (6) or (15)})
\end{aligned}$$

□

Now we show that $Ax^* = b$. Recall that $P = \{(r-1)(2c+1) + 2j' + 2 \mid r \in [m-1], j' \in \mathbb{Z}_c\}$. Let $m' = (m-1)(2c+1) + c + 1$, be the number of rows in A . To prove $Ax^* = b$, we need to show that,

$$\sum_{j=1}^{n'} A[q, j] \cdot x[j] = \begin{cases} L - 1, & \text{if } q \in P \\ 1, & \text{otherwise.} \end{cases}$$

We consider the following exhaustive cases.

Case 1: $q \in P$. Let $q = (r-1)(2c+1) + 2i + 2$ for some $r \in [m-1]$ and $i \in \mathbb{Z}_c$. Notice that $q \in I_r, q \in I_{r+1}$ and $q \notin I_{r'}$ for every $r' \in [m] \setminus \{r, r+1\}$. This implies that

$$\text{for every } j \notin J_r \cup J_{r+1}, A[q, j] = 0. \quad (18)$$

Then

$$\begin{aligned}
\sum_{j=1}^{n'} A[q, j] \cdot x[j] &= \sum_{s \in [m]} \sum_{j \in J_s} A[q, j] \cdot x[j] \\
&= \left(\sum_{j \in J_r} A[q, j] \cdot x[j] \right) + \left(\sum_{j \in J_{r+1}} A[q, j] \cdot x[j] \right) \quad (\text{By (18)}) \\
&= \left(\sum_{i' \in \mathbb{Z}_c} \sum_{j \in P_{r,i'}} A[q, j] \cdot x[j] \right) + \left(\sum_{i' \in \mathbb{Z}_c} \sum_{j \in P_{r+1,i'}} A[q, j] \cdot x[j] \right) \\
&= \left(\sum_{j \in P_{r,i}} A[q, j] \cdot x[j] \right) + \left(\sum_{j \in P_{r+1,i}} A[q, j] \cdot x[j] \right) \quad (\text{By Claims 3.2(c) and 3.3(c)}) \\
&= L - 1 - j_i + j_i \quad (\text{By Claims 3.2(a) and 3.3(a)}) \\
&= L - 1.
\end{aligned}$$

Case 2: $q \in [m'] \setminus P$. We partition $[m'] \setminus P$ into $R_1 \uplus R_2 \uplus R_3$, and consider sub-cases based on these parts. Let

$$\begin{aligned}
R_1 &= \{(r-1)(2c+1) + 2j' + 3 \mid r \in [m-1], j' \in \mathbb{Z}_c\}, \\
R_2 &= \{(r-1)(2c+1) + 1 \mid r \in [m]\}, \\
R_3 &= [(m-1)(2c+1) + 1 + c] \setminus [(m-1)(2c+1) + 1].
\end{aligned}$$

Case 2(a): $q \in R_1$. Let $q = (r - 1)(2c + 1) + 2i + 3$ for some $r \in [m - 1]$ and $i \in \mathbb{Z}_c$. Notice that $q \in I_r, q \in I_{r+1}$ and $q \notin I_{r'}$ for any $r' \in [m] \setminus \{r, r + 1\}$. This implies that,

$$\text{for any } j \notin J_r \cup J_{r+1}, A[q, j] = 0. \quad (19)$$

Hence

$$\begin{aligned} \sum_{j=1}^{n'} A[q, j] \cdot x[j] &= \sum_{s \in [m]} \sum_{j \in J_s} A[q, j] \cdot x[j] \\ &= \left(\sum_{j \in J_r} A[q, j] \cdot x[j] \right) + \left(\sum_{j \in J_{r+1}} A[q, j] \cdot x[j] \right) \quad (\text{By (19)}) \\ &= \left(\sum_{i' \in \mathbb{Z}_c} \sum_{j \in P_{r, i'}} A[q, j] \cdot x[j] \right) + \left(\sum_{i' \in \mathbb{Z}_c} \sum_{j \in P_{r+1, i'}} A[q, j] \cdot x[j] \right) \\ &= \left(\sum_{j \in P_{r, i}} A[q, j] \cdot x[j] \right) + \left(\sum_{j \in P_{r+1, i}} A[q, j] \cdot x[j] \right) \quad (\text{By Claims 3.2(c) and 3.3(c)}) \\ &= 1 + 0 \quad (\text{By Claims 3.2(b) and 3.3(b)}) \\ &= 1 \end{aligned}$$

Case 2(b): $q \in R_2$. Let $q = (r - 1)(2c + 1) + 1$ for some $r \in [m]$. By construction of A , we have that for all $j \notin J_r$, $A[q, j] = 0$. This implies that

$$\sum_{j=1}^{n'} A[q, j] \cdot x[j] = \sum_{j \in J_r} A[q, j] \cdot x[j]. \quad (20)$$

We consider two cases based on $r = 1$ or $r > 1$. When $r > 1$,

$$A[\{(r - 1)(2c + 1) + 1\}, J_r] = B_r[\{2c + 1\}, J_r].$$

Recall the function α . That is, if $\alpha(C_r) = g$, then $\phi_{j_g}(X_g)$ satisfies C_r . By Equation (20),

$$\begin{aligned} \sum_{j=1}^{n'} A[q, j] \cdot x[j] &= \sum_{j=1}^{c \cdot 2^{\ell+1}} B_r[2c + 1, j] \cdot x[j] \\ &= \sum_{i' \in \mathbb{Z}_c} \sum_{j \in [2^{\ell+1}]} B_r[2c + 1, i' \cdot 2^{\ell+1} + j] \cdot x[i' \cdot 2^{\ell+1} + j] \\ &= \sum_{j \in [2^{\ell+1}]} B_r[2c + 1, \alpha(C_r) \cdot 2^{\ell+1} + j] \cdot x[\alpha(C_r) \cdot 2^{\ell+1} + j] \\ &\quad + \sum_{i' \in \mathbb{Z}_c \setminus \{\alpha(C_r)\}} \sum_{j \in [2^{\ell+1}]} B_r[2c + 1, i' \cdot 2^{\ell+1} + j] \cdot x[i' \cdot 2^{\ell+1} + j] \\ &= \left(\sum_{j \in [2^{\ell+1}]} B_r[2c + 1, \alpha(C_r) \cdot 2^{\ell+1} + j] \cdot x[\alpha(C_r) \cdot 2^{\ell+1} + j] \right) + 0 \\ &\quad (\text{By (2) and definition of } Q) \\ &= B_r[2c + 1, \alpha(C_r) \cdot 2^{\ell+1} + 2\alpha(C_r) + 2] \\ &= 1. \quad (\text{By (3) using the fact that } \phi_{j_{\alpha(C_r)}} \text{ satisfies } C_r) \end{aligned}$$

When $r = 1$ we have that $q = 1$ and $A[\{1\}, J_1] = B_1[\{1\}, J_1]$. Hence, by Equation (20),

$$\begin{aligned}
\sum_{j=1}^{n'} A[q, j] \cdot x[j] &= \sum_{j \in 1}^{c \cdot 2^{\ell+1}} B_1[1, j] \cdot x[j] \\
&= \sum_{i' \in \mathbb{Z}_c} \sum_{j \in [2^{\ell+1}]} B_1[1, i' \cdot 2^{\ell+1} + j] \cdot x[i' \cdot 2^{\ell+1} + j] \\
&= \sum_{j \in [2^{\ell+1}]} B_1[1, \alpha(C_1) \cdot 2^{\ell+1} + j] \cdot x[\alpha(C_1) \cdot 2^{\ell+1} + j] \\
&\quad + \sum_{i' \in \mathbb{Z}_c \setminus \{\alpha(C_1)\}} \sum_{j \in [2^{\ell+1}]} B_1[1, i' \cdot 2^{\ell+1} + j] \cdot x[i' \cdot 2^{\ell+1} + j] \\
&= \left(\sum_{j \in [2^{\ell+1}]} B_1[1, \alpha(C_1) \cdot 2^{\ell+1} + j] \cdot x[\alpha(C_1) \cdot 2^{\ell+1} + j] \right) + 0 \\
&\quad \text{(By (11) and definition of } Q\text{)} \\
&= B_1[1, \alpha(C_1) \cdot 2^{\ell+1} + 2\alpha(C_1) + 2] \quad \text{(By (11) and definition of } Q\text{)} \\
&= 1 \quad \text{(By (12) using the fact that } \phi_{j_{\alpha(C_1)}} \text{ satisfies } C_1\text{)}
\end{aligned}$$

Case 2(c): $q \in R_3$. Let $q = (m-1)(2c+1) + 1 + i$ where $i \in [c]$. By the definition of A , we have that for every $j \notin \{(m-1) \cdot c \cdot 2^{\ell+1} + (i-1)2^{\ell+1} + 1, \dots, (m-1) \cdot c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1}\}$, $A[q, j] = 0$. That is, for $j \notin P_{m,i-1}$, $A[q, j] = 0$. Let $P_{m,i-1} \cap Q = \{(m-1)c \cdot 2^{\ell+1} + (i-1)2^{\ell+1} + 2j_{i-1} + g\}$, where $g \in \{1, 2\}$. Hence

$$\begin{aligned}
\sum_{j=1}^{n'} A[q, j] \cdot x[j] &= \sum_{j \in P_{m,i-1}} A[q, j] \cdot x[j] \\
&= \sum_{j \in P_{m,i-1} \cap Q} B_m[2c+1+i, j - (m-1)c2^{\ell+1}] \\
&= B_m[2c+1+i, (i-1)2^{\ell+1} + 2j_{i-1} + g] \\
&= 1 \quad \text{(By (13))}
\end{aligned}$$

□

Lemma 3.4. *The path-width of the column matroid of A is at most $c+4$*

Proof. Recall that $n' = m \cdot c \cdot 2^{\ell+1}$, be the number of columns in A and m' be the number of rows in A . To prove that the path-width of A is at most $c+4$, it is sufficient to show that for all $j \in [n'-1]$,

$$\dim(\text{span}(A[\{1, \dots, j\}]) \cap \text{span}(A[\{j+1, \dots, n'\}])) \leq c+3. \quad (21)$$

The idea for proving Equation (21) is based on the following observation. For $V' = A[\{1, \dots, j\}]$ and $V'' = A[\{j+1, \dots, n'\}]$, let

$$I = \{q \in [m'] \mid \text{there exists } v' \in V' \text{ and } v'' \in V'' \text{ such that } v'[q] \neq v''[q] \neq 0\}.$$

Then the dimension of $\text{span}(V') \cap \text{span}(V'')$ is at most $|I|$. Thus to prove (21), for each $j \in [n'-1]$, we construct the corresponding set I and show that its cardinality is at most $c+3$.

We proceed with the details. Let $v_1, v_2, \dots, v_{n'}$ be the column vectors of A . Let $j \in [n' - 1]$. Let $V_1 = \{v_1, \dots, v_j\}$ and $V_2 = \{v_{j+1}, \dots, v_{n'}\}$. We need to show that $\dim(\text{span}(V_1) \cap \text{span}(V_2)) \leq c + 3$. Let

$$I' = \{q \in [m'] \mid \text{there exists } v \in V_1 \text{ and } v' \in V_2 \text{ such that } v[q] \neq 0 \neq v'[q]\}.$$

We know that $[n']$ is partitioned into parts $P_{r', i'}, r' \in [m], i' \in \mathbb{Z}_c$.

Fix $r \in [m]$ and $i \in \mathbb{Z}_c$ such that $j \in P_{r, i}$.

Let $j = (r - 1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 2j' + g$, where $j' \in \mathbb{Z}_L$ and $g \in \{1, 2\}$. Let $q_1 = \max\{0, (r - 1)(2c + 1) - 2c\}$, $q_2 = r(2c + 1)$, $j_1 = (r - 1) \cdot c \cdot 2^{\ell+1}$, and $j_2 = r \cdot c \cdot 2^{\ell+1}$. Then $[q_2] \setminus [q_1] = I_r$ and $[j_2] \setminus [j_1] = J_r$ (recall the definition of sets I_r and J_r from the construction of matrix A).

The way we constructed matrix A , for every $q > q_2$ and for every vector $v \in V_1$, we have $v[q] = 0$. Also, for every $q \leq q_1$ and for any $v \in V_2$, we have that $v[q] = 0$. This implies that $I' \subseteq [q_2] \setminus [q_1] = I_r$. Now we partition I_r into 5 parts: R_1, R'_1, R, R_2 , and R'_2 . These parts are defined as follows.

$$\begin{aligned} R_1 &= \begin{cases} \emptyset, & \text{if } r = 1, \\ \{(r - 2)(2c + 1) + 2i' + 2 \mid i' \in \mathbb{Z}_c\}, & \text{otherwise,} \end{cases} \\ R'_1 &= \begin{cases} \emptyset, & \text{if } r = 1, \\ \{(r - 2)(2c + 1) + 2i' + 3 \mid i' \in \mathbb{Z}_c\}, & \text{otherwise,} \end{cases} \\ R &= \{(r - 1)(2c + 1) + 1\}, \end{aligned} \tag{22}$$

$$R_2 = \begin{cases} \emptyset, & \text{if } r = m, \\ \{(r - 1)(2c + 1) + 2i' + 2 \mid i' \in \mathbb{Z}_c\}, & \text{otherwise,} \end{cases}$$

and

$$R'_2 = \begin{cases} \{(r - 1)(2c + 1) + i' + 1 \mid i' \in [c]\}, & \text{if } r = m, \\ \{(r - 1)(2c + 1) + 2i' + 3 \mid i' \in \mathbb{Z}_c\}, & \text{otherwise.} \end{cases}$$

Claim 3.5. For each $r \in [m], q \notin I_r$ and $j'' \in J_r, v_{j''}[q] = 0$.

Proof. The non-zero entries in A are covered by the disjoint sub-matrices $A[I_r, J_r] = B_r, r \in [m]$. Hence the claim follows. \square

Claim 3.6. $|I' \cap R_1| \leq c - (i - 1)$.

Proof. When $r = 0$, $R_1 = \emptyset$ and the claim trivially follows. Let $r > 1$, and let $q \in R_1$ be such that $q < (r - 2)(2c + 1) + 2i + 2$. Then $q = (r - 2)(2c + 1) + 2i' + 2$ for some $i' < i$. Notice that $q \notin I_{r'}$ for every $r' > r$. By Claim 3.5, for every $v \in \bigcup_{r' > r} J_{r'}$, $v[q] = 0$. Now consider the vector $v_{j''} \in V_2 \setminus (\bigcup_{r' > r} J_{r'})$. Notice that $j'' > j$ and $j'' \in J_r$. Let $j'' = j + a = (r - 1)c \cdot 2^{\ell+1} + i \cdot 2^{\ell+1} + 2j' + g + a$ for some $a \in [rc2^{\ell+1} - j]$. From the construction of A , $v_{j''}[q] = B_r[2i' + 1, i \cdot 2^{\ell+1} + 2j' + g + a] = 0$, by (6). Thus for every $q \in R$, $q < (r - 2)(2c + 1) + 2i + 2$ and $v \in V_2$, $v[q] = 0$. This implies that

$$|I' \cap R_1| \leq |\{q \geq (r - 2)(2c + 1) + 2i + 2\} \cap R_1| \leq c - (i - 1).$$

\square

Claim 3.7. $|I' \cap R'_1| = 0$.

Proof. When $r = 1$, $R'_1 = \emptyset$ and the claim holds. So, now assume that $r > 1$. Consider any $q \in R'_1$. Let $i' \in \mathbb{Z}_c$ be such that $q = (r-2)(2c+1) + 2i' + 3$. Notice that $q \notin I_{r'}$ for any $r' > r$, and hence, by Claim 3.5, for any $v \in \bigcup_{r' > r} J_{r'}$, $v[q] = 0$. Now consider any $j'' \in J_r$. Let $j'' = (r-1)c \cdot 2^{\ell+1} + a$, for some $a \in [c \cdot 2^{\ell+1}]$. From the construction of A , $v_{j''}[q] = B_r[2i' + 2, a] = 0$, by (6) or (7). This completes the proof of the claim. \square

Claim 3.8. $|I' \cap R_2| \leq i$.

Proof. When $r = m$, $R_2 = \emptyset$ and the claim trivially holds. So, now let $r < m$ and consider any $q \in R_1 \cap \{q > (r-1)(2c+1) + 2i + 2\}$. Let $i' > i$ such that $q = (r-1)(2c+1) + 2i' + 2$. Notice that $q \notin I_{r'}$ for any $r' < r$. Hence, by Claim 3.5, for any $v \in \bigcup_{r' < r} J_{r'}$, $v[q] = 0$. Now consider any vector $v_{j''} \in V_1 \setminus (\bigcup_{r' < r} J_{r'})$. Notice that $j'' \leq j$ and $j'' \in J_r$. Let $j'' = (r-1)c \cdot 2^{\ell+1} + i'' \cdot 2^{\ell+1} + a$ for some $a \in [2^{\ell+1}]$ and $i'' \leq i < i'$. From the construction of A , $v_{j''}[q] = B_r[2c + 2i' + 2, i'' \cdot 2^{\ell+1} + a] = 0$, by (8). Hence we have shown that for any $q \in R$, $q > (r-2)(2c+1) + 2i + 2$ and $v \in V_1$, $v[q] = 0$. This implies that

$$|I' \cap R_2| \leq |\{q \leq (r-1)(2c+1) + 2i + 2\} \cap R_1| \leq i$$

\square

Claim 3.9. $|I' \cap R'_2| \leq 1$.

Proof. Consider the case when $r = m$. Consider $q \in R'_2$. We claim that if $q \in I' \cap R'_2$, then $q = (m-1)(2c+1) + i + 2$. Suppose $q \in I' \cap R'_2$ and $q < (m-1)(2c+1) + i + 2$. Let $q = (m-1)(2c+1) + i' + 2$, where $0 \leq i' < i$. Then by the construction of A , for any $j'' > j$, $v_{j''}[q] = B_m[2c + i' + 2, j'' - (m-1)c2^{\ell+1}] = B_m[2c + i' + 2, i_1 2^{\ell+1} + a]$, where $c-1 \geq i_1 \geq i$ and $a \in [2^{\ell+1}]$. Thus by (17), $V_{j''}[q] = B_m[2c + i' + 2, i_1 2^{\ell+1} + a] = 0$. This contradicts the assumption that $q \in I' \cap R'_2$.

Suppose that $q \in I' \cap R'_2$ and $q > (m-1)(2c+1) + i + 2$. Let $q = (m-1)(2c+1) + i' + 2$, where $i < i' \leq c-1$. Then by the construction of A , for any $j'' \leq j$, $v_{j''}[q] = B_m[2c + i' + 2, j'' - (m-1)c2^{\ell+1}] = B_m[2c + i' + 2, i_1 2^{\ell+1} + a]$, where $0 \leq i_1 \leq i$, $a \in [2^{\ell+1}]$. Thus by (17), $v_{j''}[q] = B_m[2c + i' + 2, i_1 2^{\ell+1} + a] = 0$. This contradicts the assumption that $i \in I' \cap R'_2$. Hence, in this case, we have proved that $|I' \cap R'_2| \leq 1$.

So, now assume that $r \in [m-1]$. Consider any $q \in R'_2$. Let $i' \in \mathbb{Z}_c$ such that $q = (r-1)(2c+1) + 2i' + 3$. Notice that $q \notin I_{r'}$ for any $r' < r$, and hence, by Claim 3.5, for any $v \in \bigcup_{r' < r} J_{r'}$, $v[q] = 0$. Also notice that $q \notin I_{r'}$ for any $r' > r+1$, and hence, by Claim 3.5, for any $v \in \bigcup_{r' > r+1} J_{r'}$, $v[q] = 0$. So the only potential j'' for which $v_{j''}[q] \neq 0$, are from $J_r \cup J_{r+1}$. Let $j'' \in J_{r+1}$. Then by the definition of A , $v_{j''}[q] = B_{r+1}[2i' + 2, j'' - rc2^{\ell}] = 0$, by (6) or (7) or (15) or (16). Hence, we conclude that the only possible j'' for which $v_{j''}[q] \neq 0$, are from J_r .

Now the proof is similar to case when $r = m$. We claim that if $q \in I' \cap R'_2$, then $q = (r-1)(2c+1) + 2i + 3$. Suppose $q \in I' \cap R'_2$ and $q < (r-1)(2c+1) + 2i + 3$. Let $q = (r-1)(2c+1) + 2i' + 3$, where $0 \leq i' < i$. Then by the construction of A , for any $j'' > j$, $v_{j''}[q] = B_r[2c + 2i' + 3, j'' - (r-1)c2^{\ell+1}] = B_r[2c + 2i' + 3, i_1 2^{\ell+1} + a]$, where $c-1 \geq i_1 \geq i$ and $a \in [2^{\ell+1}]$. Thus by (8), $v_{j''}[q] = B_r[2c + 2i' + 3, i_1 2^{\ell+1} + a] = 0$. This contradicts the assumption that $q \in I' \cap R'_2$.

Suppose $q \in I' \cap R'_2$ and $q > (r-1)(2c+1) + 2i + 3$. Let $q = (r-1)(2c+1) + 2i' + 3$, where $i < i' < c$. Then by the construction of A , for any $j'' \leq j$, $v_{j''}[q] = B_r[2c + 2i' + 3, j'' - (r-1)c2^{\ell+1}] = B_r[2c + 1 + i', i_1 2^{\ell+1} + a]$, where $0 \leq i_1 \leq i$, $a \in [2^{\ell+1}]$. Thus by (8), $v_{j''}[q] = B_r[2c + 2i' + 3, i_1 2^{\ell+1} + a] = 0$. This contradicts the assumption that $i \in I' \cap R'_2$. Hence, in this case as well, $|I' \cap R'_2| \leq 1$. This completes the proof of the claim. \square

Therefore, we have

$$\begin{aligned}
|I'| &= |I' \cap I_r| && \text{(Because } I' \subseteq I_r \text{)} \\
&= |I' \cap R_1| + |I' \cap R'_1| + |I' \cap R| + |I' \cap R_2| + |I' \cap R'_2| \\
&&& \text{(By (22) and Claims 3.6, 3.7, 3.8 and 3.9)} \\
&\leq c - (i - 1) + 0 + 1 + i + 1 \\
&= c + 3
\end{aligned}$$

This completes the proof of the lemma. \square

Proof of Theorem 1.4. We prove the theorem by assuming a fast algorithm for (IP) and use it to give a fast algorithm for CNF-SAT, refuting SETH. Let ψ be an instance of CNF-SAT with n_1 variables and m_1 clauses. We choose a sufficiently large constant c such that $(1 - \epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c} < 1$ holds. We use the reduction mentioned in Lemma 3.1 and construct an instance $A_{(\psi, c)}x = b_{(\psi, c)}, x \geq 0$, of (IP) which has a solution if and only if ψ is satisfiable. The reduction takes time $\mathcal{O}(m_1^2 2^{\frac{n_1}{c}})$. Let $\ell = \lceil \frac{n_1}{c} \rceil$. The constraint matrix $A_{(\psi, c)}$ has dimension $((m_1 - 1)(2c + 1) + 1 + c) \times (m_1 \cdot c \cdot 2^{\ell+1})$ and the largest entry in vector $b_{(\psi, c)}$ does not exceed $2^\ell - 1$. The path-width of $M(A_{(\psi, c)})$ is at most $c + 4$.

Assuming that any instance of (IP) with non-negative constraint matrix of path-width k is solvable in time $f(k)(d + 1)^{(1-\epsilon)k}(mn)^a$, where d is the maximum value in an entry of b and $\epsilon, a > 0$ are constants, we have that $A_{(\psi, c)}x = b_{(\psi, c)}, x \geq 0$, is solvable in time

$$f(c + 4) \cdot 2^{\ell \cdot (1-\epsilon)(c+4)} \cdot 2^{\ell \cdot a} \cdot m_1^{\mathcal{O}(1)} = 2^{\frac{n_1}{c}(1-\epsilon)(c+4)} \cdot 2^{\frac{n_1 \cdot a}{c}} \cdot m_1^{\mathcal{O}(1)} = 2^{n_1((1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c})} \cdot m_1^{\mathcal{O}(1)}.$$

Here the constant $f(c + 4)$ is subsumed by the term $m_1^{\mathcal{O}(1)}$. Hence the total running time for testing whether ψ is satisfiable or not, is,

$$\mathcal{O}(m_1^2 2^{\frac{n_1}{c}}) + 2^{n_1((1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c})} m_1^{\mathcal{O}(1)} = 2^{n_1((1-\epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c})} m_1^{\mathcal{O}(1)} = 2^{\epsilon' \cdot n_1} m_1^{\mathcal{O}(1)},$$

where $\epsilon' = (1 - \epsilon) + \frac{4(1-\epsilon)}{c} + \frac{a}{c} < 1$. This completes the proof of Theorem 1.4. \square

4 Proof Sketch of Theorem 1.5

In this section we prove that (IP) with non-negative matrix A cannot be solved in time $f(d)(d + 1)^{(1-\epsilon)k}(mn)^{\mathcal{O}(1)}$ for any function f and $\epsilon > 0$, unless SETH fails, where $d = \max\{b[1], \dots, b[m]\}$ and k is the path-width of the column matroid of A .

In Section 3, we gave a reduction from CNF-SAT to (IP). However in this reduction the values in the constraint matrix $A_{(\psi, c)}$ and target vector $b_{(\psi, c)}$ can be as large as $2^{\lceil \frac{n}{c} \rceil} - 1$, where n is the number of variables in the CNF-formula ψ and c is a constant. Let m be the number of clauses in ψ . In this section we briefly explain how to get rid of these large values, at the cost of making large, but still bounded path-width. From a CNF-formula ψ , we construct a matrix $A = A_{(\psi, c)}$ as described in Section 3. The only rows in A which contain values strictly greater than 1 (values other than 0 or 1) are the rows indexed from the set $P = \{(r - 1)(2c + 1) + 1 + 2i + 1 \mid r \in [m - 2], i \in \mathbb{Z}_c\}$. That is the values greater than 1 are in the alternate rows in yellow/green colored portion except the last c rows in A in Figure 2b. Recall that $\ell = \lceil \frac{n}{c} \rceil$ and the largest value in A is $2^\ell - 1$. Any number less than or equal to $2^\ell - 1$ can be represented by a binary string of length $\ell = \frac{n}{c}$. Now we construct a new matrix A' from A by replacing each row of A whose index is in the set P with ℓ rows and

for any value $A[i, j], i \in P$ we write its ℓ -bit binary representation in the column corresponding to j and the newly added ℓ rows of A' . That is, for any $\gamma \in P$, we replace the row γ with ℓ rows, $\gamma_1, \dots, \gamma_\ell$. Where, for any j , if the value $A[\gamma, j] = W$ then $A'[\gamma_k, j] = \eta_k$, where η_k is the k^{th} bit in the ℓ -sized binary representation of W .

Let m' be the number of rows in A' . Now the target vector b' is defined as $b'[i] = 1$ for all $i \in [m']$. This completes the construction of the reduced (IP) instance $A'x = b'$. The correctness proof of this reduction is using arguments similar to those used for the correctness of Lemma 3.1.

Lemma 4.1. *The path-width of the column matroid of A' is at most $(c+1)\frac{n}{c} + 3$.*

Proof. We sketch the proof, which is similar to the proof of Lemma 3.4. We define I'_r and J'_r for any $r \in [m]$ like I_r and J_r in Section 3. In fact, the rows in I'_r are the rows obtained from I_r in the process explained above to construct A' from A . We need to show that $\dim(\text{span}(A'|\{1, \dots, j\}) \cap \text{span}(A'|\{j+1, \dots, n'\})) \leq (c+1)\frac{n}{c} + 2$ for all $j \in [n'-1]$, where n' is the number of columns in A' . The proof proceeds by bounding the number of indices I such that for any $q \in I$ there exist vectors $v \in A'|\{1, \dots, j\}$ and $u \in A'|\{j+1, \dots, n'\}$ with $v[q] \neq 0 \neq u[q]$. By arguments similar to the ones used in the proof of Lemma 3.4, we can show that for any $j \in [n'-1]$, the corresponding set I' of indices is a subset of I'_r for some $r \in [m]$. Recall the partition of I_r into R_1, R'_1, R, R_2 and R'_2 in Lemma 3.4. We partition I'_r into parts S_1, S'_1, S, S_2 and S'_2 . Here $S'_1 = R_1, S = R$ and $S'_2 = R_2$. Notice that $R_1, R_2 \subseteq P$, where P is the set of rows which covers all values strictly greater than 1. The set S_1 and S_2 are obtained from R_1 and R_2 , respectively, by the process mentioned above to construct A' from A . That is, each row in $R_i, i \in \{1, 2\}$ is replaced by ℓ rows in S_i . This allows us to bound the following terms for some $i \in \mathbb{Z}_c$:

$$\begin{aligned} |I' \cap S_1| &\leq (c - (i - 1))\ell = (c - (i - 1))\ell, \\ |I' \cap S'_1| &= 0, \\ |I' \cap S_2| &\leq i \cdot \ell, \\ |I' \cap S'_2| &\leq 1, \text{ and} \\ |I' \cap S| &\leq |S| = 1, \end{aligned}$$

By using the fact that $I' \subseteq I'_r$ and the above system of inequalities, we can show that

$$\dim(\text{span}(A'|\{1, \dots, j\}) \cap \text{span}(A'|\{j+1, \dots, n'\})) \leq (c+1)\lceil \frac{n}{c} \rceil + 2.$$

This completes the proof of the lemma. □

Now the proof of the theorem follows from Lemma 4.1 and the correctness of the reduction (it is similar to the arguments in the proof of Theorem 1.4).

5 Proof of Theorem 1.3

In this section, we sketch how the proof of Cunningham and Geelen [8] of Theorem 1.2, can be adapted to prove Theorem 1.3. Recall that a path decomposition of width k can be obtained in $f(k) \cdot n^{\mathcal{O}(1)}$ time for some function f by making use of the algorithm by Jeong et al. [23]. However, we do not know if such a path decomposition can be constructed in time $\mathcal{O}((d+1)^{k+1})n^{\mathcal{O}(1)}$, so the assumption that a path decomposition is given is essential.

Roughly speaking, the only difference in the proof is that when parameterized by the branch-width, the most time-consuming operation is the “merge” operation, when we have to construct a

new set of partial solutions with at most $(d+1)^k$ vectors from two already computed sets of sizes $(d+1)^k$ each. Thus to construct a new set of vectors, one has to go through all possible pairs of vectors from both sets, which takes time roughly $(d+1)^{2k}$. For path-width parameterization, the new partial solution set is constructed from two sets, but this time one set contains at most $(d+1)^k$ vectors while the second contains at most $d+1$ vectors. This allows us to construct the new set in time roughly $(d+1)^{k+1}$.

Recall that for $X \subseteq [n]$, we define $S(A, X) = \text{span}(A|X) \cap \text{span}(A|E \setminus X)$, where $E = [n]$. The key lemma in the proof of Theorem 1.2 is the following.

Lemma 5.1 ([8]). *Let $A \in \{0, 1, \dots, d\}^{m \times n}$ and $X \subseteq [n]$ such that $\lambda_{M(A)}(X) = k$. Then the number of vectors in $S(A, X) \cap \{0, \dots, d\}^m$ is at most $(d+1)^{k-1}$.*

To prove Theorem 1.3, without loss of generality, we assume that the columns of A are ordered in such a way that for every $j \in [n-1]$,

$$\dim(\text{span}(A|\{1, \dots, i\}) \cap \text{span}(A|\{i+1, \dots, n\})) \leq k-1.$$

Let $A' = [A, b]$. That is A' is obtained by appending the column-vector b to the end of A . Then for each $i \in [n]$,

$$\dim(\text{span}(A'|\{1, \dots, i\}) \cap \text{span}(A'|\{i+1, \dots, n+1\})) \leq k. \quad (23)$$

Now we use dynamic programming to check whether the following conditions are satisfied. For $X \subseteq [n+1]$, let $\mathcal{B}(X)$ be the set of all vectors $b' \in \mathbb{Z}_{\geq 0}^m$ such that

- (1) $0 \leq b' \leq b$,
- (2) there exists $z \in \mathbb{Z}_{\geq 0}^{|X|}$ such that $(A'|X)z = b'$, and
- (3) $b' \in S(A', X)$.

Then (IP) has a solution if and only if $b \in \mathcal{B}([n])$. Initially the algorithm computes for all $i \in [n]$, $\mathcal{B}(\{i\})$ and by Lemma 5.1, we have that $|\mathcal{B}(\{i\})| \leq d+1$. In fact $\mathcal{B}(\{i\}) \subseteq \{a \cdot v \mid v \text{ is the } i^{\text{th}} \text{ column vector of } A' \text{ and } a \in [d+1]\}$. Then for each $j \in [2, \dots, n]$ the algorithm computes $\mathcal{B}([j])$ in increasing order of j and outputs YES if and only if $b \in \mathcal{B}([n])$. That is, $\mathcal{B}([j])$ is computed from the already computed sets $\mathcal{B}([j-1])$ and $\mathcal{B}(\{j\})$. Notice that $b' \in \mathcal{B}([j])$ if and only if

- (a) there exist $b_1 \in \mathcal{B}(\{1, \dots, j-1\})$ and $b_2 \in \mathcal{B}(\{j\})$ such that $b' = b_1 + b_2$,
- (b) $b' \leq b$ and
- (c) $b' \in S(A', [j])$.

So the algorithm enumerates vectors b' satisfying condition (a), and each such vector b' is included in $\mathcal{B}([j])$, if b' satisfy conditions (b) and (c). Since by (23) and Lemma 5.1, $|\mathcal{B}([j-1])| \leq (d+1)^k$ and $|\mathcal{B}(\{j\})| \leq d+1$, the number of vectors satisfying condition (a) is $(d+1)^k$, and hence the exponential factor of the required running time follows. This provides the bound on the claimed exponential dependence in the running time of the algorithm. The bound on the polynomial component of the running time follows from exactly the same arguments as in [8].

6 Proof of Theorem 1.1

In this section we prove that unless ETH fails, (IP) cannot be solved in time $n^{o(\frac{m}{\log m})}d^{o(m)}$, where $d = \max\{b[1], \dots, b[m]\}$, even when the constraint matrix is non-negative and all entries in any feasible solution is at most 2.

Our proof is by a reduction from a 3-CNF SAT to (IP). From a 3-CNF formula ψ on n variables and m clauses we create an equivalent (IP) instance $A_\psi x = b_\psi, x \geq 0$, where A_ψ is a non-negative integer matrix of order $(2m+n) \times (2(m+n))$ and the largest entry in b_ψ is 3. Our reduction work in polynomial time. Let ψ be the input of 3-CNF SAT. Let $X = \{x_1, \dots, x_n\}$ be the set of variables in ψ and $\mathcal{C} = \{C_1, \dots, C_m\}$ be the set of clauses in ψ . Now we create $2n+2m$ number of 0–1 vectors of length $2m+n$, two per variable and two per clause. For each $x_i \in X$ we make two vectors v_{x_i} and $v_{\bar{x}_i}$. They are defined as follows. For $j \in [m]$, we set

$$v_{x_i}[j] = \begin{cases} 1 & \text{if } x_i \in C_j, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$v_{\bar{x}_i}[j] = \begin{cases} 1 & \text{if } \bar{x}_i \in C_j, \\ 0 & \text{otherwise.} \end{cases}$$

For $j = m+i$, we put $v_{x_i}[j] = v_{\bar{x}_i}[j] = 1$ and for all $j \in \{m+1, \dots, 2m+n\} \setminus \{m+i\}$, we define $v_{x_i}[j] = v_{\bar{x}_i}[j] = 0$.

For every clause $C_j \in \mathcal{C}$, we define two vectors v_{C_j} and $v_{C'_j}$ as follows. For $i \in [m]$, we define

$$v_{C_j}[i] = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \quad (24)$$

and

$$v_{C'_j}[i] = 0.$$

For $i = m+n+j$, we set $v_{C_j}[i] = v_{C'_j}[i] = 1$. For every $m+n+j \neq i \in \{m+1, \dots, 2m+n\}$, we put $v_{C_j}[i] = v_{C'_j}[i] = 0$.

Matrix A_ψ is constructed using these vectors as columns. The columns of A_ψ are ordered as

$$v_{x_1}, v_{\bar{x}_1}, v_{x_2}, v_{\bar{x}_2}, \dots, v_{x_n}, v_{\bar{x}_n}, v_{C_1}, v_{C'_1}, v_{C_2}, v_{C'_2}, \dots, v_{C_m}, v_{C'_m}.$$

Vector b_ψ is defined as follows.

$$b_\psi[i] = \begin{cases} 3 & \text{if } i \in [m], \\ 1 & \text{if } i \in [m+n] \setminus [m], \\ 2 & \text{if } i \in [2m+n] \setminus [m+n]. \end{cases}$$

Lemma 6.1. *Formula ψ is satisfiable if and only if $A_\psi x = b_\psi, x \geq 0$ is feasible.*

Proof. Suppose that the formula ψ is satisfiable and let ϕ be a satisfying assignment of ψ . We define a $(2n+2m)$ -vector x^* and prove that $A_\psi x^* = b_\psi$. For any $i \in \mathbb{Z}_n$,

$$x^*[2i+1] = \begin{cases} 1 & \text{if } \phi(x_{i+1}) = 1, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$x^*[2i+2] = \begin{cases} 1 & \text{if } \phi(\bar{x}_{i+1}) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

This completes the definition of first $2n$ entries of x^* . The other $2m$ entries (the last $2m$ entries) of x^* is defined as follows. For every $i \in \mathbb{Z}_m$, we define

$$x^*[2n+2i+1] = \begin{cases} 0 & \text{if the number of literals set to 1 in } C_{i+1} \text{ by } \phi \text{ is 3,} \\ 1 & \text{if the number of literals set to 1 in } C_{i+1} \text{ by } \phi \text{ is 2,} \\ 2 & \text{otherwise,} \end{cases} \quad (25)$$

and

$$x^*[2n+2i+2] = \begin{cases} 2 & \text{if the number of literals set to 1 in } C_{i+1} \text{ by } \phi \text{ is 3,} \\ 1 & \text{if the number of literals set to 1 in } C_{i+1} \text{ by } \phi \text{ is 2,} \\ 0 & \text{otherwise.} \end{cases} \quad (26)$$

We now proceed to prove that x^* is indeed a feasible solution.

Claim 6.2. $A_\psi x^* = b_\psi$.

Proof. Towards this, we need to show that for every $i \in [2m+n]$, $\sum_{j=1}^{2n+2m} A_\psi[i, j] \cdot x^*[j] = b_\psi[i]$. We consider the following exhaustive cases.

Case $i \in [m]$. The fact that each clause has 3 literals along with the definition of A_ψ implies that the number of entries set to 1 in $\{A_\psi[i, j] \mid j \in [2n]\}$ is 3. Also, the indices j for which $A_\psi[i, j]$ is set to one, correspond to a literal in C_i . By the definition of x^* and the fact that ϕ is a satisfying assignment, we have that $\sum_{j=1}^{2n} A_\psi[i, j]x^*[j] \in [3]$. Let $r = \sum_{j=1}^{2n} A_\psi[i, j]x^*[j]$. Hence

$$\begin{aligned} \sum_{j=1}^{2n+2m} A_\psi[i, j]x^*[j] &= \left(\sum_{j=1}^{2n} A_\psi[i, j]x^*[j] \right) + \left(\sum_{j=2n+1}^{2n+2m} A_\psi[i, j]x^*[j] \right) \\ &= r + \left(\sum_{j=2n+1}^{2n+2m} v_{C_i}[j]x^*[j] \right) \\ &= r + v_{C_i}[i]x^*[2n+2(i-1)+1] \quad (\text{By (24) and construction of } A_\psi) \\ &= r + 1 \cdot (3-r) \quad (\text{By (24) and (25)}) \\ &= 3 = b_\psi[i]. \end{aligned}$$

Case $i \in [m+n] \setminus [m]$. By the definition of A_ψ and vectors $v_{x_j}[i], v_{\bar{x}_j}[i], j \in [2n]$, we have that $A_\psi[i, j] = 1, j \in [2n]$ if and only if $j \in \{2(i-1)+1, 2(i-1)+2\}$. By the definition of x^* , exactly one from $\{x^*[2(i-1)+1], x^*[2(i-1)+2]\}$ is set to 1. This implies that $\sum_{j=1}^{2n} A_\psi[i, j]x^*[j] = 1$. By the construction of A_ψ , we have that $A_\psi[i, j] = 0$ for every $j \in [2m+2n] \setminus [2n]$. Therefore, $\sum_{j=1}^{2n+2m} A_\psi[i, j]x^*[j] = 1 = b_\psi[i]$

Case $i \in [2m+n] \setminus [m+n]$. Let $i = m+n+i'$. From the construction A_ψ , we have that $A_\psi[i, j]$ is set to zero for all $j \in [2n]$, and for any $j \in [2m+2n] \setminus [2n]$, $A_\psi[i, j]$ is set to 1 if and only if $j \in \{2n+2(i'-1)+1, 2n+2(i'-1)+2\}$. This implies that

$$\begin{aligned} \sum_{j=1}^{2n+2m} A_\psi[i, j]x^*[j] &= x^*[2n+2(i'-1)+1] + x^*[2n+2(i'-1)+2] \\ &= 2 = b_\psi[i] \quad (\text{By (25) and (26)}) \end{aligned}$$

This completes the proof of the claim. \square

For the converse direction of the statement of the lemma, suppose that there exists $x^* \in \mathbb{Z}_{\geq 0}^{2n+2m}$ such that $A_\psi x^* = b_\psi$. Now we need to show that ψ is satisfiable. We first argue that for any $i \in [n]$, exactly one of $\{x^*[2(i-1)+1], x^*[2(i-1)+2]\}$ is set to 1 with the other set to 0. This follows from the fact that (i) $A_\psi[m+i, 2(i-1)+1] = A_\psi[m+i, 2(i-1)+2] = 1$, (ii) for all $j \in [2n+2m] \setminus \{2(i-1)+1, 2(i-1)+2\}$, $A_\psi[m+i, j] = 0$ and (iii) $b_\psi[m+i] = 1$. Now we define an assignment ϕ and prove that ϕ is a satisfying assignment for ψ . For $i \in [n]$ we define

$$\phi(x_i) = \begin{cases} 1 & \text{if } x^*[2(i-1)+1] = 1, \\ 0 & \text{if } x^*[2(i-1)+2] = 1. \end{cases}$$

We claim that ϕ satisfies all the clauses. Consider a clause C_j where $j \in [m]$. Since $A_\psi[m+n+j, 2n+2(j-1)+1] = 1$, $b_\psi[m+n+j] = 2$, and x^* is a feasible solution, we have that

$$A_\psi[m+n+j, 2n+2(j-1)+1] \cdot x^*[2n+2(j-1)+1] \leq b_\psi[m+n+j] = 2.$$

This implies that $x^*[2n+2(j-1)+1] \leq 2$. Let $C_j = x \vee y \vee z$ where $x, y, z \in \{x_i, \bar{x}_i \mid i \in [n]\}$. Notice that from the construction A_ψ there are 3 distinct columns $i_x, i_y, i_z \in [2n]$ such that i_w^{th} column of A_ψ is same as the vector v_w , where $w \in \{x, y, z\}$. From the construction of A_ψ , the only non-zero entries in row numbered j are $A_\psi[j, i_x], A_\psi[j, i_y], A_\psi[j, i_z]$ and $A_\psi[j, 2n+2(j-1)+1]$. We have proved that $x^*[2n+2(j-1)+1] \leq 2$ and notice that $b_\psi[j] = 3$. This implies that at least one among $\{x^*[i_x], x^*[i_y], x^*[i_z]\}$ is 1 and the corresponding entry in row j is 1. This implies that ϕ satisfies C_j . This completes the proof of the lemma. \square

Now we show that for every feasible solution x , the largest entry in x is at most 2. Notice that $b_\psi[i] = 3$ for all $i \in [m]$ and $b_\psi[i] < 3$ for all $i \in [2m+n] \setminus [m]$. This implies that for any feasible solution x , $x[i] \leq 2$ for all $i \in [2m+n] \setminus [m]$. From the construction of A_ψ we have that for $A_\psi[i, j] \neq 0, i \in [m]$, there exists an $i' \in [2m+n] \setminus [m]$ such that $A_\psi[i', j] = 1$. This along with the fact that $b_\psi[i'] < 3$ implies that in every feasible solution x , $x[i] \leq 2$ for all $i \in [2m+n] \setminus [m]$. Hence the largest entry in any feasible solution is at most 2. The following lemma completes the proof of the theorem.

Lemma 6.3. *If there is an algorithm for (IP) runs in time $n^{o(\frac{m}{\log m})} d^{o(m)}$, where $d = \max\{b[1], \dots, b[m]\}$, then ETH fails.*

Proof. By the sparsification lemma [22], we know that 3-CNF SAT on n' variables and cn' clauses, where c is a constant, cannot be solved in time $2^{o(n')}$ time. Suppose there is an algorithm **ALG** for (IP) running in time $n^{o(\frac{m}{\log m})} d^{o(m)}$. Then for a 3-CNF formula ψ with n' variables and $m' = cn$ clauses we create an instance $A_\psi x = b_\psi, x \geq 0$ of (IP) as discussed in this section, in polynomial time, where A_ψ is a matrix of dimension $(2cn' + n') \times (2(n' + cn'))$ and the largest entry in b_ψ is 3. The rank of A_ψ is at most $(2cn' + n')$. Then by Lemma 6.1, we can run **ALG** to test whether ψ is satisfiable or not. This takes time

$$(2(cn' + n'))^{o(\frac{2cn' + n'}{\log(2cn' + n')})} \cdot 3^{o(2cn' + n')} = 2^{o(n')},$$

hence refuting ETH. \square

7 Conclusion

We conclude with several open questions. First of all, while our SETH-based lower bounds for (IP) with non-negative constraint matrix are tight for path-width parameterization, there is a “ $(d+1)^k$

to $(d+1)^{2k}$ gap” between lower and upper bounds for branch-width parameterization. Closing this gap is the first natural question.

The proof of Cunningham-Geelen of Theorem 1.2 consists of two parts. The first part bounds the number of potential partial solutions corresponding to any edge of the branch decomposition tree by $(d+1)^k$. The second part is the dynamic programming over the branch decomposition using the fact that the number of potential partial solutions is bounded. The bottleneck in Cunningham-Geelen’s algorithm is the following subproblem. We are given two vector sets A and B of partial solutions, each set of size at most $(d+1)^k$. We need to construct a new vector set C of partial solutions, where the set C will have size at most $(d+1)^k$ and each vector from C is the *sum* of a vector from A and a vector from B . Thus to construct the new set of vectors, one has to go through all possible pairs of vectors from both sets A and B , which takes time roughly $(d+1)^{2k}$.

A tempting approach towards improving the running time of this particular step could be the use of *fast subset convolution* or *matrix multiplication* tricks, which work very well for “join” operations in dynamic programming algorithms over tree and branch decompositions of graphs [14, 30, 13], see also [11, Chapter 11]. Unfortunately, we have reason to suspect that these tricks may *not* help for matrices: solving the above subproblem in time $(d+1)^{(1-\epsilon)2k}n^{\mathcal{O}(1)}$ for any $\epsilon > 0$ would imply that 3-SUM is solvable in time $n^{2-\epsilon}$, which is believed to be unlikely. (The 3-SUM problem asks whether a given set of n integers contains three elements that sum to zero.) Indeed, consider an equivalent version of 3-SUM, named 3-SUM’, which is defined as follows. Given 3 sets of integers A, B and C each of cardinality n , and the objective is to check whether there exist $a \in A, b \in B$ and $c \in C$ such that $a + b = c$. Then, 3-SUM is solvable in time $n^{2-\epsilon}$ if and only if 3-SUM’ is as well (see Theorem 3.1 in [17]). However, the problem 3-SUM’ is equivalent to the most time consuming step in the algorithm of Theorem 1.2, where the integers in the input of 3-SUM’ can be thought of as length-one vectors. While this observation does not *rule out* the existence of an algorithm solving (IP) with constraint matrices of branch-width k in time $(d+1)^{(1-\epsilon)2k}n^{\mathcal{O}(1)}$, it indicates that any interesting improvement in the running time would require a completely different approach.

Our final open question is to obtain a refined lower bound for (IP) with bounded rank. Recall that the constraint matrix of the algorithm of Papadimitriou [26] can contain *negative* values and improving the running time of his algorithm or showing that its running time is tight up to SETH, is still a very interesting question.

References

- [1] A. ABBOUD, A. BACKURS, AND V. V. WILLIAMS, *Tight hardness results for LCS and other sequence similarity measures*, in Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2015, pp. 59–78. [3](#)
- [2] A. ABBOUD AND V. V. WILLIAMS, *Popular conjectures imply strong lower bounds for dynamic problems*, in Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2014, pp. 434–443. [3](#)
- [3] A. BACKURS AND P. INDYK, *Edit distance cannot be computed in strongly subquadratic time (unless SETH is false)*, in Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC), ACM, 2015, pp. 51–58. [3](#)
- [4] —, *Which regular expression patterns are hard to match?*, in Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2016, p. to appear. [3](#)
- [5] K. BRINGMANN, *Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails*, in Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2014, pp. 661–670. [3](#)
- [6] K. BRINGMANN AND M. KÜNNEMANN, *Quadratic conditional lower bounds for string problems and dynamic time warping*, in Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2015, pp. 79–97. [3](#)
- [7] W. COOK AND P. D. SEYMOUR, *Tour merging via branch-decomposition*, *INFORMS Journal on Computing*, 15 (2003), pp. 233–248. [1](#)
- [8] W. H. CUNNINGHAM AND J. GEELEN, *On integer programming and the branch-width of the constraint matrix*, in Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO), vol. 4513 of Lecture Notes in Comput. Sci., Springer, 2007, pp. 158–166. [1](#), [20](#), [21](#)
- [9] R. CURTICAPEAN AND D. MARX, *Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus*, in Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2016, pp. 1650–1669. [3](#)
- [10] M. CYGAN, H. DELL, D. LOKSHTANOV, D. MARX, J. NEDERLOF, Y. OKAMOTO, R. PATURI, S. SAURABH, AND M. WAHLSTRÖM, *On problems as hard as CNF-SAT*, in Proceedings of the 27th IEEE Conference on Computational Complexity (CCC), IEEE, 2012, pp. 74–84. [3](#)
- [11] M. CYGAN, F. V. FOMIN, L. KOWALIK, D. LOKSHTANOV, D. MARX, M. PILIPCZUK, M. PILIPCZUK, AND S. SAURABH, *Parameterized Algorithms*, Springer, 2015. [3](#), [25](#)
- [12] M. CYGAN, S. KRATSCH, AND J. NEDERLOF, *Fast hamiltonicity checking via bases of perfect matchings*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC), ACM, 2013, pp. 301–310. [3](#)
- [13] M. CYGAN, J. NEDERLOF, M. PILIPCZUK, M. PILIPCZUK, J. M. M. VAN ROOIJ, AND J. O. WOJTASZCZYK, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2011, pp. 150–159. [25](#)

- [14] F. DORN, *Dynamic programming and fast matrix multiplication*, in Proceedings of the 14th Annual European Symposium on Algorithms (ESA), vol. 4168 of Lecture Notes in Comput. Sci., Springer, Berlin, 2006, pp. 280–291. [25](#)
- [15] F. V. FOMIN, P. A. GOLOVACH, D. LOKSHTANOV, AND S. SAURABH, *Almost optimal lower bounds for problems parameterized by clique-width*, SIAM J. Computing, 43 (2014), pp. 1541–1563. [3](#)
- [16] F. V. FOMIN AND D. M. THILIKOS, *Dominating sets in planar graphs: Branch-width and exponential speed-up*, SIAM J. Computing, 36 (2006), pp. 281–309. [1](#)
- [17] A. GAJENTAN AND M. H. OVERMARS, *On a class of $o(n^2)$ problems in computational geometry*, Comput. Geom., 5 (1995), pp. 165–185. [25](#)
- [18] P. HLINĚNÝ, *Branch-width, parse trees, and monadic second-order logic for matroids*, J. Combinatorial Theory Ser. B, 96 (2006), pp. 325–351. [1](#)
- [19] P. HLINĚNÝ, *The tutte polynomial for matroids of bounded branch-width*, Combinatorics, Probability & Computing, 15 (2006), pp. 397–409. [1](#)
- [20] G. B. HORN AND F. R. KSCHISCHANG, *On the intractability of permuting a block code to minimize trellis complexity*, IEEE Trans. Information Theory, 42 (1996), pp. 2042–2048. [2](#)
- [21] R. IMPAGLIAZZO AND R. PATURI, *On the complexity of k -SAT*, J. Computer and System Sciences, 62 (2001), pp. 367–375. [1](#), [3](#)
- [22] R. IMPAGLIAZZO, R. PATURI, AND F. ZANE, *Which problems have strongly exponential complexity*, J. Computer and System Sciences, 63 (2001), pp. 512–530. [3](#), [24](#)
- [23] J. JEONG, E. J. KIM, AND S. OUM, *Constructive algorithm for path-width of matroids*, in Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2016, pp. 1695–1704. [4](#), [20](#)
- [24] D. LOKSHTANOV, D. MARX, AND S. SAURABH, *Known algorithms on graphs on bounded treewidth are probably optimal*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2011, pp. 777–789. [3](#)
- [25] D. LOKSHTANOV, D. MARX, AND S. SAURABH, *Lower bounds based on the exponential time hypothesis*, Bulletin of the EATCS, 105 (2011), pp. 41–72. [3](#)
- [26] C. H. PAPADIMITRIOU, *On the complexity of integer programming*, J. ACM, 28 (1981), pp. 765–768. [1](#), [2](#), [25](#)
- [27] M. PĂTRAȘCU AND R. WILLIAMS, *On the possibility of faster SAT algorithms*, in Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2010, pp. 1065–1075. [3](#)
- [28] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. X. Obstructions to tree-decomposition*, J. Combinatorial Theory Ser. B, 52 (1991), pp. 153–190. [1](#), [4](#)
- [29] L. RODITTY AND V. V. WILLIAMS, *Fast approximation algorithms for the diameter and radius of sparse graphs*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC), ACM, 2013, pp. 515–524. [3](#)

- [30] J. M. M. VAN ROOIJ, H. L. BODLAENDER, AND P. ROSSMANITH, *Dynamic programming on tree decompositions using generalised fast subset convolution*, in Proceedings of the 17th Annual European Symposium on Algorithms (ESA), vol. 5757 of Lecture Notes in Comput. Sci., Springer, 2009, pp. 566–577. [25](#)
- [31] V. V. WILLIAMS, *Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk)*, in Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC), vol. 43 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2015, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 17–29. [3](#)